AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 1/3 DESIGN AND IMPLEMENTATION OF A MICROPROCESSOR FLIGHT DAMAGE MON--ETC(U) DEC 79 B C JOHNSON AD-A080 417 AFIT/GCS/EE/79-6 NL UNCLASSIFIED 1 04 2 AC 208041 *

DESIGN AND IMPLEMENTATION OF A

MICROPROCESSOR FLIGHT DAMAGE MONITOR.

(9) Mister's THESIS,

Charles

(14) AFIT/GCS/EE/79-6

70 Brian Johnson USAF

(1) Der 17

(B) 190 /

FE.B

690

DISTRIBUTION STATEMENT A

Approved for public released Distribution Unlimited

6/2275

Mit

DESIGN AND IMPLEMENTATION OF A MICROPROCESSOR FLIGHT DAMAGE MONITOR

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University (ATC)
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science

bу

Brian C. Johnson Capt USAF

Graduate Computer Systems

December 1979

A

Approved for public release; distribution unlimited.

Preface

This thesis describes the design and implementation of a microprocessor based data acquisition and processing system for the Air Force Flight Dynamics Laboratory. Wright-Patterson Air Force Base Ohio. A structured design of the system was performed, and a bench model system was implemented. The bench model demonstrated that realtime data acquisition and computation of a cumulative damage matrix can be performed.

I wish to thank my advisor, Dr. Gary B. Lamont for his help during the design and implementation of this system, and for help in preparing this report. I also would like to thank the members of my thesis committee, Dr. Thomas C. Hartrum and Dr. John M. Borky, for their help with this thesis. For their assistance and support. I would like to thank my sponsors Dr. Lynn C. Rogers and Mr. Robert W. Gordon, of the Air Force Flight Dynamics Laboratory; and for his help with the software development. I would like to thank Mr. Frank Beitel of the Air Force Materials Laboratory Computer Activities Division, Wright-Patterson Air Force Base, Ohio.

Last but not least, I would like to express my indebtedness to my wife Cynthia, for her love, patience, and help in preparing this thesis.

Brian C Johnson

CONTENTS

	·	
Prefa	e	l 1
List	of Figures	11
List	of Tables · · · · · · · · · · · · · · · · · · ·	Ĺχ
Abst	ct	x
1.	Introduction	1
	Background	2
	System Concept · · · · · · · · · · · · · · · · · · ·	4
	Approach	5
	Development Environment	5
	Scope of Thesis	7
11.	Requirements Definition	9
	Introduction · · · · · · · · · · · · · · · · · · ·	9
	System Requirements	9
	-	12
III.	Algorithm Design	3
	Introduction · · · · · · · · · · · · · · · · · · ·	13
		13
	Data Processing Algorithm	3
	FFT (Fast Fourier Transform)	4
	POWERSPECTRUM	9
		0
		21
		21
		•
	Configuration Algorithm	2
		23
		4
		6
		6
	_	6
		27
		7
	· · · · · · · · · · · · · · · · · · ·	?7
		8
	PWRUPDN (Power-Up-Down)	8
		n

IV.	Hardware Considerations
	Introduction
	LSI-11 Microcomputer System
	KD11-J Processor Module
	MRV11-B UV-PROM
	MMV11-A Core Memory
	ADVII-A Analog-to-Digital Converter 3
	KNVll-A Programmable Real Time Clock 3
	DLV11-J Serial Line Interface
	DDV11-B Backplane
	IBVII-A LSI-II/Instrument Bus Interface 30
	KPV11-B Power Fail/Line Time Clock/Terminator 30
	Analog Sensors
	System Clock
	Summary
v.	Software Design
	Introduction
	Program Language Decision
	Configuration Modules
	MFDM (Microprocessor Flight Damage Monitor) 49
	SETUP
	GETFREQUENCY
	GETTIME
	GETDATE
	GETCOMIENT
	ERROR
	CDFOUTPUT (Cumulative Damage Factor Output) 53
	Execution Modules
	XECUTE (Execute)
	PWRUPDN (Power-Up-Down)
	Data Processing Modules
	EXECUTIVE
	FFT (Fast Fourier Transform)
	POWERSPECTRUM
	DAMAGEFACTOR
	Summary
VI.	Testing and Experimental Results 68
	Introduction
	Testing of Software Modules 68
	Integration of the Hardware and the Software 69
	Bench Model Configuration 69
	Bench Model Testing 69
	Experimental Results
	Explanation of CDF Table
	Same and

VII.	Prop	osed	Fligh	t-Wo	rt	hy	S	ys	te	11	•	•	•	•	•	•	•	•	•	•	•	•	•	78
	In	trodu	ction			•	•	•		•			•		•					.•			•	78
	Ha	rdwar	e			•	•	•	•			•	•		•				•					78
	So	ftware	e				•	•	•					•	•	•	•				٠.		•	80
	Po	wer R	equir	emen	ts			•	•	٠														83
		ckagi																						85
		mary		• •																				85
VIII.	Reco	mmenda	ation	s an	d	Co	nc:	lus	sic	วนะ	9	•	•	•	•	•	•	•	•	•	•	•	•	89
	In	troduc	ction			•	•	•			•		•						•					89
		comme																						89
		nclus																						90
Bibliogr	aphy	• • •			•	•		•	•	•	•		•	•	•	•	•	•		•	•	•		92
Appendix	. A:	FORTI	RAN S	OUTC	1 م	Cod	đe	AT	'n	F) Ot	. . (Ch	a r	te									
appendan		for (93
		101 (JOHL I	guta		JII	1110	Juc	116	= 3	•	•	•	٠	•	•	•	•	•	•	•	•	•	73
		MFI	om .		•	•	•	•	•	•	•	•	•	•	•	•		•	•	•	•	•	•	94
		ERI	ROR .		•	•	•	•	•	•	•	•	٠	•	•	•	•		•		•	•	•	98
		SET	rup .		•												•	•						100
		GE?	FREO	UENC	Y	•																		194
		GE?	TIME																					108
			DATE																					110
			12:001																					112
			POUTP																					114
Appendix	В٠	FORTE	RAN S	ourc	e (: :0	ie.	ar	ı d	FI	O	s (?h	ari	te	f	ar.							
		Data																		_	_		_	119
					6	٠.			•••		\		•••	.			٠		•	•	•	•	•	
		EX	ECUTI	VE		•	•	•	٠	•	•	•	•	•	•		•	•	•	•	•	•	•	120
		FF7	٠ 1		•	•	•		•	•	٠	•	•	•	•		•	•		•	•	•	•	133
		PO	PERSPI	ECTR	MU	•					٠			•	•						٠			155
		DA!	IACEF	ACTO	R																			158
																								160
		PO	ERUP																					163
Appendix	c:	Users	' Mar	nual	•	•	•	•	•	•	•	•	•	•	•		•	•	•		•	•	•	167
		Int	roduc	ctio	n			•																167
		Phy	sica:	L Co	nfi	gı	ıra	ti	.on	ì	•		•	•	•	•	•	•	•	•	•	•	•	167
		Sys	ten S	Star	t -1	1p	Pr	oc	ed	lut	·e			•		•								170
			ratio																					171
										-	_	,		-	•	•	•	-	-		•	•	-	
		S	ETUP	• •	•	•	•	•	٠	•	•	٠	•	٠	•	٠	٠	•	•	•	•	•	•	171
		-	ATE		•	•	•	•	•	•	٠	•	•	•	•	•	•	•	•	•	•	•	•	172
			OMME							•	•	٠	•	•	•	٠	•	•	•	•	•	•	•	172
		F	REQUI	ency	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	173
		P	UN DI	JRAT	ION	ł	•	•	•	•	•	•		٠	•		•	•	•			•	•	173
		E	XECUT	re .		•	•		•		•				•						•	•	•	174
		^	ilim Dila																					17/

Stopping	P	ro	gı	at	n 1	Exc	ec	ut:	101	n	•	•	•	•	•	•	•	•	•	•	175
Summary .						•	•	•				•	•								175

List of Figures

Figur	ಆ	Page
1	Damage Factor versus Frequency	2
2	Cumulative Damage Matrix	4
3	Activity Diagram for the MFDM	6
4	Steps in Computing the Cumulative Damage Factor	15
5	Data Flow Diagram for Data Processing	16
6	Structure Chart for Data Processing Modules	17
7	Structure Chart for Configuration Modules	25
8	Structure Chart for the MFDM System	29
9	Temperature Signal versus Temperature	38
10	Structure Chart for the MFDM System	44
11	Bench Model System Configuration	70
12	Proposed System Configuration	79
13	Flow Chart for Control of Auto-Ranging Gain	82
14	Proposed Concept of the NFDM	86
15	Subunit Layout for Proposed System	87
16	Flow Chart for MFDM	94
17	Flow Chart for ERROR	98
18	Flow Chart for SETUP	100
19	Flow Chart for GETFREQUENCY	104
20	Flow Chart for GETTIME	108
21	Flow Chart for GETDATE	110
22	Flow Chart for GETCOMMENT	112
23	Flow Chart for CDFOUTPUT	114
24	Flow Chart for EXECUTIVE	120
0.5	When the sale from white	100

26	Flow Chart for POWERSPECTRUM	155
27	Flow Chart for DAMAGEFACTOR	158
28	Flow Chart for XECUTE	160
29	Flow Chart for POWERUPDN	163
30	Block Diagram of Bench Model System	168
31	Bench Model Backplane Configuration	169

List of Tables

Table		Page
I	LSI-11 Support Hardware	7
11	The Required Frequency Ranges and Intervals	11
III	Nyquist Sampling Rate for the Required Frequency Ranges	20
IA	Interfaces for Data Processing Modules	22
V	Gain Settings for Frequency Amplifier	39
VI	Specifications for A2583 Automatic Gain Ranging Amplifier	39
VII	Arguments of the MFDM Modules	45
VIII	Initialized Constants Used in FFT	64
IX	Cumulative Danage Output Table	73
X	Specifications for Proposed Modules	84

Abstract

In order to design a constrained layer damping treatment for aircraft components—the Air Force Flight Dynamics Laboratory at Uright-Patterson Air Force Base—Ohio requires a knowledge of vibration and temperature variations which a component encounters during its life. This report discusses the design and implementation of a microprocessor based system to acquire vibration and temperature information, and computes the cumulative damage factor versus frequency and temperature in real time. The cumulative damage factor is computed for one of eight user selectable frequency ranges and for a temperature range of -50 to +225 degrees Fahrenheit—and is stored in a non-volatile memory.

The designed system uses an LSI II microcomputer system consisting of an LSI-II processor, an analog-to-digital converter, a real-time clock. PROM, a non-volatile memory, and a serial interface. The software developed for this system controls the data acquisition, the computation of a fast Fourier transform (FFT), the power spectrum of vibration data, and the subsequent computation of the cumulative damage factor. The data acquisition and FFT modules were coded in assembly language, and the other modules were coded in FORTRAN. The output of the system consists of a table containing the cumulative damage as a function of frequency and temperature. A bench model system was constructed and tested, and a proposal for a flight-worthy system is made

DESIGN AND IMPLEMENTATION OF A MICROPROCESSOR FLIGHT DAMAGE MONITOR

I. Introduction

Aircraft and avionics hardware operate in an environment of varying temperature and vibration. The resonant fatigue damage caused by this environment is a major factor in airframe and avionics hardware structural failure. In order to reduce the resonant fatigue damage, design engineers must know what temperatures and vibration frequencies an object will encounter during its life. With this information, a constrained layer damping treatment which reduces the resonant fatigue damage can be applied to the object (Ref 10).

The Air Force Flight Dynamics Laboratory (AFFDL), Wright-Patterson Air Force Base Ohio, has the technology to produce a constrained layer damping treatment, although the current flight data acquisition and analysis systems are very large and hard to use, and require undue time and resources (Ref 10). In order to simplify the data acquisition, a microprocessor based data acquisition and processing system to acquire the necessary data, and process it in real-time, was designed (Ref 7). This investigation, sponsored by AFFDL, is a continuation of that effort and will focus on an expanded design, bench model construction, testing, and recommendations for a flight-worthy system.

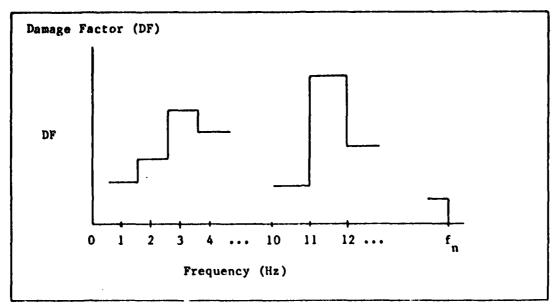


Fig 1. Damage Factor versus Frequency

Background

Research has determined that the resonant vibrations which cause structural damage occur at frequencies under 6,000 Hz and in a temperature range between -50 degrees Fahrenheit and +225 degrees Fahrenheit (Ref 10). The damage factor (DF) at a given frequency (f) and temperature (T) can be approximated with the following equation (Ref 10):

$$DFT(f) = K(\overline{A})^B/(f)^{2B-1}, \qquad (1)$$

where k is a constant, A is the root mean square (RMS) value of the acceleration at the specified frequency (f), and B is approximately

3.3. Figure 1 is a graph of damage factor vs. frequency computed by using Eq 1. This damage factor, when computed for a given frequency

and temperature, and summed for a predetermined period of time (t), forms the cumulative damage factor (CDF) as shown in Eq 2:

$$CDF(f,T) = \sum_{t} DF_{t}(f,T)$$
, (2)

The cumulative damage factor for a given frequency and temperature forms one damage coefficient in the "three dimensional" cumulative damage matrix, as shown in Figure 2 (Ref 10).

Current in-flight data acquisition techniques use analog signal conditioners and multi-channel analog tape recorders to record the frequency and temperature data necessary to compute the CDF. After the data has been acquired, the CDF is computed and plotted. This technique requires skilled personnel to operate the equipment, and due to the time required to compute the cumulative damage factor, there can be a long delay between the ending of the test and obtaining the cumulative damage matrix output (Ref 10).

In order to simplify this process, it was decided to design a Microprocessor Flight Damage Monitor (MFDM) which would acquire and process the data in real-time. Such a system appeared attractive, as it would be easy to install and remove, would not require skilled personnel to operate, and the cumulative damage data would be available immediately after the test (Ref 10).

System Concept

The MFDM system concept consists of a self-contained configuration which would compute, in real time, the cumulative damage factor CDF from analog temperature and frequency data, and store the cumulative

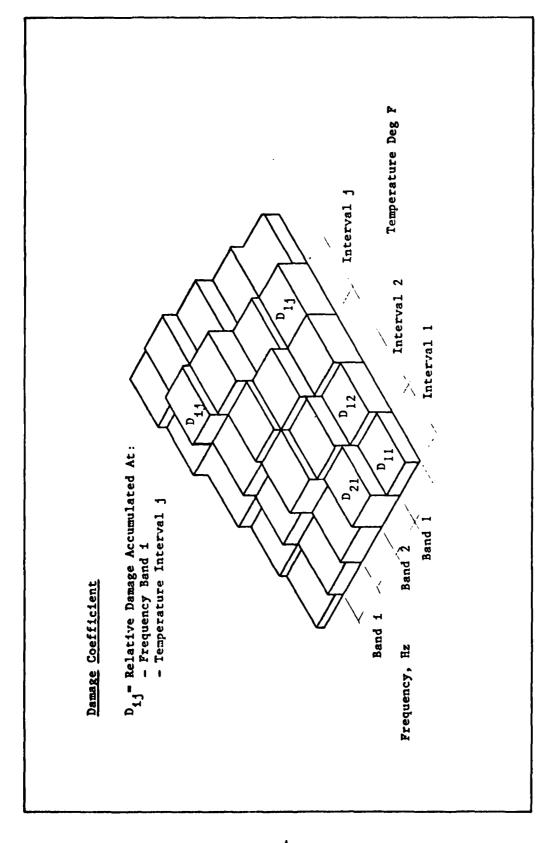


Fig 2. Cumulative Damage Matrix

damage factor in non-volatile memory. This concept was expanded to include an interactive capability where the user can select the frequency range and resolution for which the CDF is to be computed, select the duration of the test, and have the cumulative damage matrix printed (Ref 10). In addition to the interactive requirements, two start-up modes were required. The first mode would start processing data upon system power initiation and the second mode would start processing data when given a start command. Figure 3 shows an activity diagram (Ref 7 11) which summarizes the system.

Approach

The MFDM (also referred to as the "system") was developed in four phases bassed upon execpted top down structured approach. The four phases were: the conceptual phase the design phase, the implementation phase, and the testing phase. The conceptual phase uses structured analysis and design to define the detailed requirements placed on the system. The design phase included the algorithm design and selection of the system hardware. During the implementation phase, the system is coded, assembled, and given a functional check. Then the testing phase verifies the operation of the system.

Development Environment

An LSI-11 microcomputer, manufactured by Digital Equipment

Corporation (DEC), was used as the system processor because of its

powerful instruction set, processing speed, data structure and

availability of support hardware and software. The support hardware

used in the MFDM is listed in Table I. In addition to the LSI-11

microcomputer and support hardware, the development system had a

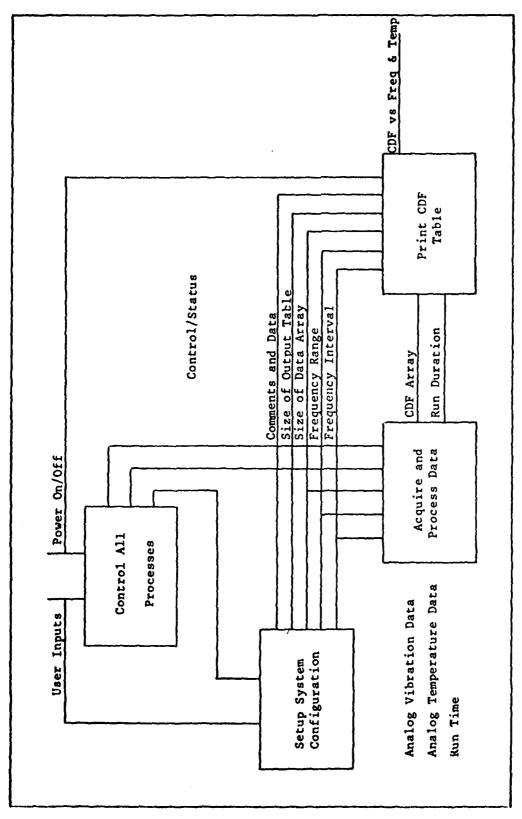


Fig 3. Activity Diagram for the MFDM

A Charles of Marie 1

TABLE I
LSI-11 Support Hardware

	Module	Description
1.	KD11-J	LSI-11 Processor
2.	MPIVII-A	4 K by 16 bit Core Hemory
3.	DLV11-J	4 Channel Asynchronous Serial Line Interface
4.	KWV11-A	Programmable Real-Time Clock
5.	ADV11-A	16 Channel 12 bit Analog-to- Digital Converter
6.	MSV11-DB	8 K by 16 bit Semiconductor Memory (Used Two Modules)
i.	BCV1B	Backplane Expansion Cables
8.	H9270	Backplanes (two)
9.	TEVII	120 OHM Bus Terminator
	н780-н	Powersupply

Heathkit H27 dual floppy disk drive, a Heathkit H14 line printer, and a Lear Siegler ADM-3 terminal. The software was developed using DEC's RT-11 version 03-B operating system, version 2.1-11 FORTRAN IV compiler, and version 03-01 macroassembler (Ref 8:643-647). This development system was located at the Air Force Institute of Technology Wright-Patterson AFB, Ohio.

Scope of Thesis

First the system requirements and design constraints are defined (Chap II), and then a structured analysis and design of the system is performed, producing a structure chart for the entire system (Chap III). Following this, the hardware selections (Ref 7 25-36) are reviewed with respect to the new system design (Chap IV). After the hardware considerations the required software modules are developed by modifying existing modules (Ref 7:75-95) where possible, and creating new modules for functions which had not yet been implemented (Chap V). The hardware

and the software were then integrated into a bench model of the MFDM which was functionally tested (Chap VI). A proposed flight-worthy system is then discussed (Chap VII). Recommendations for the flight-worthy system and investigative conclusions about this effort are presented in Chap VIII.

II. Requirements Definition

Introduction

Before any system can be designed and implemented, the system requirements must be defined. These requirements form a base line from which a system can be designed, and must include the purpose of the system, its input parameters, output parameters, any time constraints, and any hardware constraints. The purpose of this chapter is to accurately define the system in terms of the requirements established by the AFFDL (Ref 10) and the design constraints of the system.

System Requirements

The purpose of the MFDM is the real-time computation of the CDF as a function of frequency and temperature from analog frequency and temperature signals. The output of this system will be a cumulative damage table which contains the CDF as a function of frequency and temperature. This gives a general definition of the system, the specific requirements of which are as follows (Ref 7:6-22):

- The input to the system will be an analog frequency signal and an analog temperature signal. Signal amplifiers will amplify the signals so as to make the signals compatible with an analog-to-digital converter.
- The temperature range for which resonant fatigue damage usually occurs is between -50 degrees and +225 degrees Fahrenheit. In order to produce a damping layer, the cumulative damage factor must be known at 25 degree intervals for this temperature range.
- The frequency range over which damage occurs depends upon the object being tested as well as its environment. For these reasons, the frequency range and interval for which the DCF will be computed must be user selectable. Table II gives the frequency ranges/intervals for which the CDF must be computed in order to provide the user with the

capability of selecting the combination which best meets his needs.

- This system must be capable of operating for the duration of a transport aircraft's mission. The duration of these missions can be as long as 20 hours, so the MFDM must be capable of operating for 20 hours. Because the length of test missions vary the user must be able to set the desired run time before the mission.
- The user must be able to interactively, from a computer terminal, configure the system as desired. The interactive aspects of the system must be very easy to understand since they may be used by people without a computer background. Thus, the user should be prompted for responses and must be able to respond with simple responses such as one letter or one numeral. If a wrong input is entered by the user an appropriate error message will be written at the terminal, and the prompt line repeated.
- The user must be able to enter a test date and a line of comments into the computer. This date and comment is to be printed in the heading of the CDF output table along with the total time the system processed data. This will provide the information needed for correlating the output tables with the test mission.
- There must be two modes of operation: one, an automatic mode where, upon application of system power, data processing starts automatically for the last, user set, system configuration; and a second manual mode by which the user can start data processing by giving a start command from a terminal. This will allow the system to be used automatically when on a plane, and manually when used in a laboratory.
- When data processing has been performed for the desired length of time, data processing will automatically stop. Program control will return to the user if operating in the manual mode and will halt if operating in the automatic mode. This will prevent the system from processing data after a mission is over, and will allow the user to control the length of time for which data is processed.
- When processing data, the CDF matrix must be updated as fast as possible. CDF update rate of once every 6 seconds is desired as this will provide at least five updates during a plane's 30-second take-off roll.

TABLE II

The Required Frequency Ranges and Intervals

Frequency Range (0 - MAXFREO)	Interval
(K Hz)	(Hz)
0 - 6.0	100
0 - 3 0	100
0 - 3.0	50
0 - 3.0	25
0 - 1.5	100
0 - 1-5	50
0 - 1.5	25

- The dynamic range of the CDF matrix must be at least 360 db, which allows any two elements of the matrix to differ by 10 . The reason for the large dynamic range is that if there is a large cumulative damage factor at one frequency its large magnitude will not cause smaller cumulative damage factors to be lost due to the finite word size of the computer.
- The system program and CDF matrix must be stored in a non-volatile memory so program execution can start automatically upon power-up, and system power can be turned off after the processing is complete, without the loss of the CDF matrix.
- When an automatic power-up occurs the CDF matrix will be cleared only if it is the first power-up since a manual power-up. This allows data to be acquired from multiple flights if a manual power-up is not performed between flights.

This completes the list of system requirements defined for this project. Two other system requirements exist but are not requirements for this phase of the project (Ref 10). They are: the system must be battery-powered, and it must be as small as possible. This will allow the system to be used in a small space where no power is available such as in the wing of an airplane. Engineers at AFFDL will design the case for the system as well as the battery pack required to power the

system when an operational bench model system has been produced and tested.

Summary

This chapter has defined the general system requirements. These requirements will now be used to design algorithms which will be used to implement the system. The design and implementation of the algorithm is the subject of the following chapters

III. Algorithm Design

Introduction

This chapter explains the design of the algorithms which were used to implement the system according to the requirements defined in Chapter II. These algorithms were developed using a combination of structured design techniques (Ref 9), resulting in a structure chart which describes the system. The structure chart and associated module interface table developed in this chapter, along with the discussion of the function of each module which make up the structure chart define the algorithm.

Algorithm Development

The algorithm for this system was developed in three steps.

First, an algorithm was developed for the "data processing" function.

Then one was developed for the interactive user "configuration"

function, and finally the two algorithms were integrated into one

"system" algorithm.

Data Processing Algorithm

The basic steps which must be performed to compute the cumulative damage factor are as follows (Ref 10):

Obtain analog vibration and temperature signals from the item being tested and digitize the signals (subroutine EXECUTIVE).

Compute the Fourier transform of the vibrational data for the frequency range and interval of interest (subroutine FFT).

Compute the average power in each frequency interval (subroutine POMERSPECTRIM).

Compute the damage factor for each frequency interval (subroutine DAMAGEFACTOR).

Repeat these steps as long as desired, summing the damage factor into a cumulative damage matrix which contains the cumulative damage versus frequency and temperature.

These five steps are illustrated in Figure 4. The linear amplifiers shown in the figure are used to amplify the voltage signal of the sensors to a level required by the analog-to-digital converter.

In order to show the relationship between the modules discussed in the following paragraphs, a dataflow diagram and structure chart for the process just described is shown in Figures 5 and 6 respectively. These two figures are constructed from data obtained from Ref 7:55-65 which discusses a basic system to compute the cumulative damage factor. The next step is to define the functions of each of the modules shown in Figure 6. The Fourier transform module (Ref 7:80-85), will be discussed first, followed by the power spectrum and damage factor modules. After completing this, the module interfaces will be discussed.

FFT. This module is responsible for the computation of the fast Fourier transform (FFT) of the digitized vibration data. It converts the data from a function of time to a function of frequency. Since the input data is a sampled version of a continuous time domain signal, a discrete Fourier transform method has to be used. Eq 3 gives an expression for the transform of N sampled points from the time domain to the frequency domain.

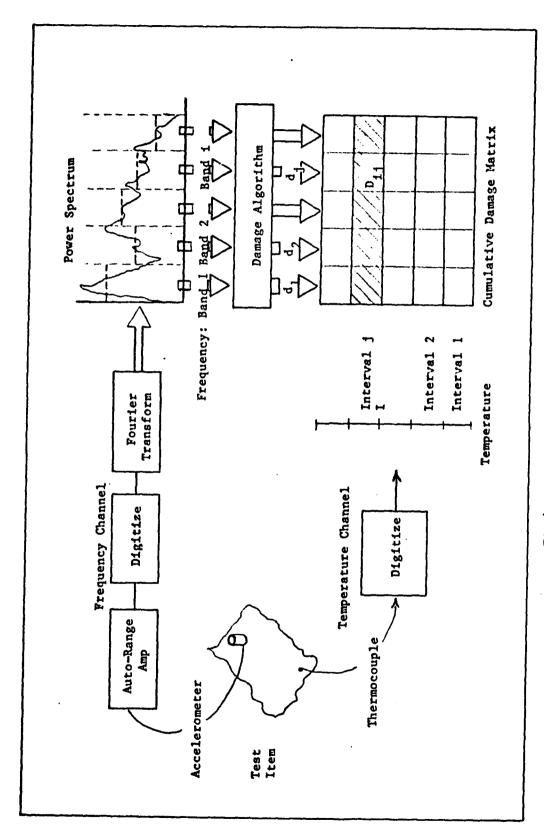


Fig 4. Steps In Computing The CDF

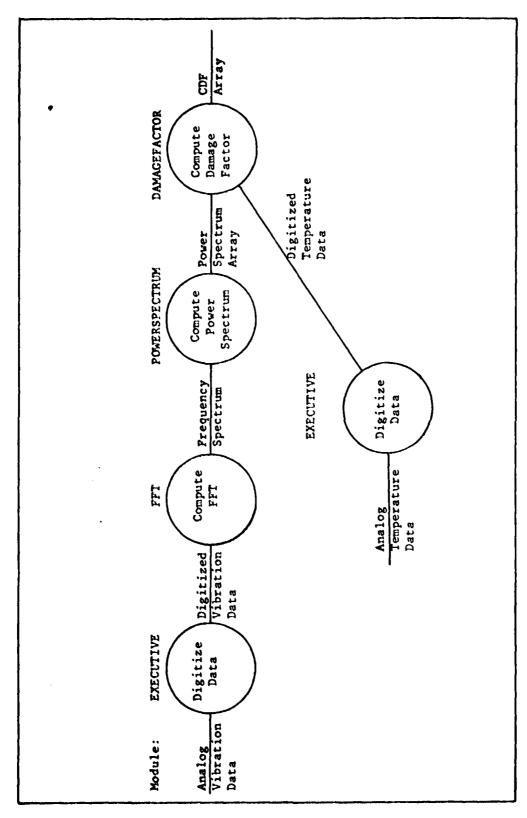


Fig 5. Data Flow Diagram for Data Processing

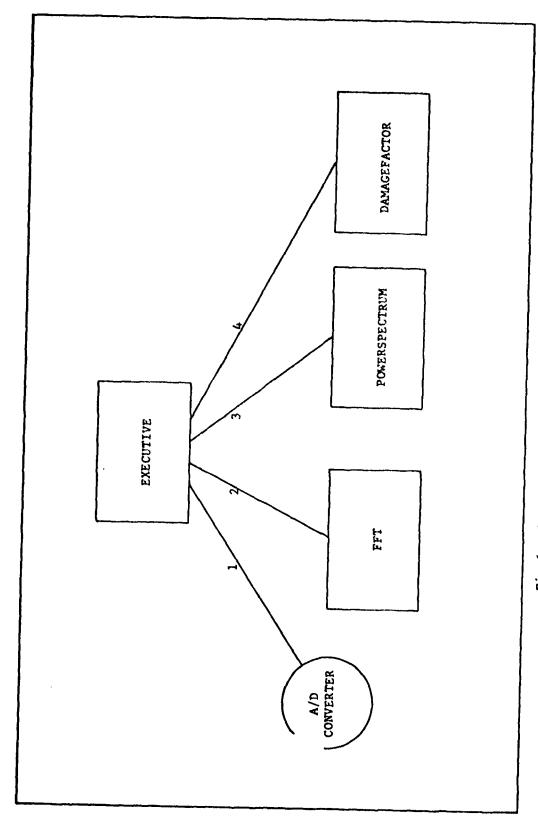


Fig 6. Structure Chart for Data Processing

Equation 3 is:

$$F(n) = \sum_{k=0}^{k=N-1} X(k) \tilde{W}^{k} , \qquad n = 0 \ 1, 2, ... N-1$$
 (3)

where

$$W = \exp(-2 i/N) \tag{4}$$

F(n) is the nth point in the violatin, and X(k) is the kth point in the time domain. ...(k) may be real or complex, but F(n) is always complex (Ref 7:70). Constraints on Eq 3 require that the frequency function be band limited at some frequency f_c , and in order to insure that no aliasing will occur in the frequency domain, the time domain function must be sampled at a rate at least twice f_c . The frequency $2f_c = 1/T$ is known as the Nyquist sampling rate, where T is the sampling interval (Ref 3 85). The number of sampled points (N) can be determined using Eq 5:

$$N\Delta f = 1/T = 2f_{c}$$
 (5)

where Δf is the desired frequency interval in the frequency domain and 1/T is the Nyquist sampling frequency (Ref 3:85-96).

Solving Eq 3 with a computer is a slow and time-consuming process as it requires on the order of N² complex multiplications and additions (Ref 4:316). To speed up this calculation, Cooley and Tukey developed a fast Fourier transform for N being an integer power of two. This reduced the order to Nx complex multiplications and

additions where x is the integer power of two, satisfying N=2^x (Ref 3:174). Since X(k) in Eq 3 can be real or complex, it is possible to solve Eq 3 with N complex data points and have the N complex data points consist of a 2N point real data sequence (Ref 3:167). This reduces the order of complexity to Nx/2. Thus Eq 3 can be efficiently implemented on a computer for N=2^x where x is an integer and the number of real data points is 2N. A complete explanation of this procedure can be found in Ref 3 148-238 and Ref 7 141-156. Table III shows the values of 2N and 1/T which satisfy these equations, using the requirements for the frequency range and interval defined in Chapter II. After completing the Fourier transform of the input data, the power within each frequency interval must be calculated.

<u>POWERSPECTRUI</u>. This module computes the power in each of the frequency intervals which make up the frequency spectrum. The frequency component F(n) resulting from the Fourier transform is a complex number which can be represented as given in Eq 6:

$$F(n) = X_n^2 + iY_n^2$$
 , $n = 0.1, 2, ... N-1$ (6)

and the power within a frequency interval can be computed using Eq 7:

$$A(n) = (X_n^2) + (Y_n^2)$$
, $n = 0, 1, 2... N-1$ (7)

Because the input signal is random, Eq 7 represents an instantaneous power, and that which is needed is the average

Table III

Nyquist Sampling Rate for the Required Frequency Ranges

Frequency Range (0 - MAXFREQ K Hz)	Interval (Hz)	2N	Sampling Rate (1/T K Hz)
0 - 6.0	100	128	12.8
0 - 3.0	100	64	6.4
0 - 3.0	50	128	6.4
0 - 3 0	25	256	6.4
0 - 1.5	100	32	3.2
0 - 1.5	50	64	3 - 2
0 - 1.5	25	128	3.2

power (Ref 10). The average power (\overline{A}) can be approximated by taking the average of a set of P instantaneous power calculations (Ref 7.73). Mathematically, this can be expressed as:

$$\overline{A}(n) = 1/P \sum_{k=1}^{P-1} A_k(n)$$
, $n = 0, 1, 2, ..., N-1$ (8)

where the subscript k represents a unique set of N instantaneous power calculations. The average power computed using Eq 8 can be interpreted as the average (RMS) value of the acceleration at the center of the frequency interval, which is the frequency of F(n). After completing the power calculation, the damage factor can be calculated.

<u>DAMAGEFACTOR</u>. The damage factor (DF) can be computed using Eq 1, which is repeated here:

$$DF(f(n)) = K(\overline{A}(n))^{3/3}/f(n)^{5/4}$$
, $n = 1.2, 3, \dots, N-1$ (1)

where k is a constant for all frequency f(n) and B in Eq 1 was replaced with its value of 3.3. Since the damage factor is a relative quantity (Ref 10), and the K factor is in all n damage factor computations it can be factored out as in Eq 9.

The cumulative damage factor (CDF) is then the sun of the damage factors calculated using Eq 9, for a period of time T. This is expressed mathematically as:

CDF(f(n) T) =
$$\sum_{t} DF_{t}(f(n))$$
, n = 1.2,3,...,N-1 (10)

where t is the rate at which the DF (f) is to be calculated, and T is the temperature interval at which the damage occurred.

This completes the discussion of the data processing modules. The remaining module, EXECUTIVE is the module which controls the data processing.

vibration data points and the temperature data, and coordinate the computation of the cumulative damage factor. The first time this module is called, it must clear the cumulative damage matrix before the data processing can begin. In addition, it must monitor the system time, and stop data processing when the users set run time has expired.

Modules' Interfaces. The next step in developing the algorithm is to establish the interfaces between the modules. Table IV lists the

Table IV

Interfaces for Data Processing Modules

INPUT	ОИТРИТ
ADDATA	ADDATA, TEMP FFTDATA
FFTDATA PWRDATA, TEMP	PWRDATA CDFMATRIX

input and output interfaces corresponding to the numbered lines in Figure 6. The executive module must acquire the 2N data points (ADDATA) and pass them to the FFT module which returns the transformed data points as N complex frequency points (FFTDATA). These complex frequency points are then passed to the power spectrum module which computes the power in each interval of the frequency spectrum. The N real power values are then returned in the array PWRDATA. The damage factor module uses PWRDATA to compute the damage factor which is then summed into the cumulative damage factor matrix (CDFMATRIX). The cumulative damage matrix is the final product, and it has a maximum size of 12 N elements, where the 12 is the number of temperature intervals required (Ref Chap II).

This completes the design of the data processing algorithm required to satisfy the requirements defined in Chapter II. The next section will discuss the design of the "configuration" algorithm which satisfies the interactive requirements defined in Chapter II.

Configuration Algorithm

A "configuration" algorithm was developed which meets the interactive requirements defined in Chapter II. This initializes the

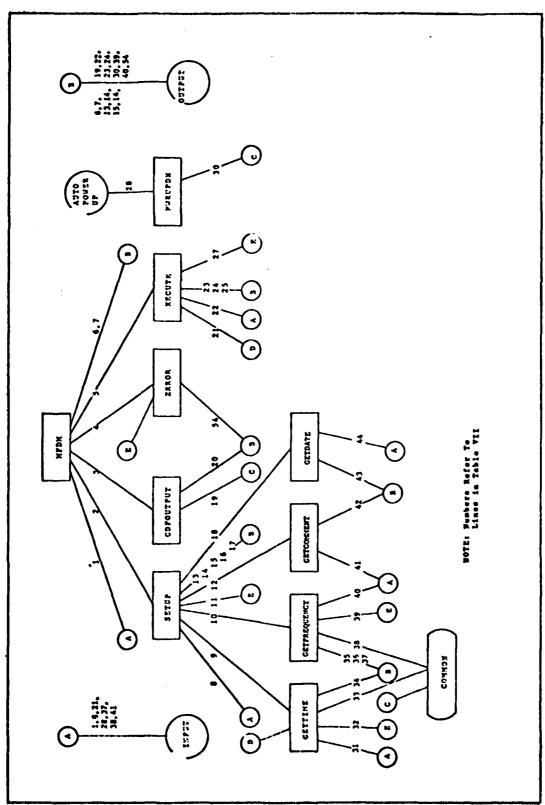
number of data points (N) which the data processing algorithm
requires and also initializes the rate at which the analog vibration
signal is to be sampled. The requirements defined in Chapter II
specify that the user be able to interactively do the following:
configure the system to process data for a specified frequency range
interval, and time period; initiate data processing; and have the CDF
matrix printed. The requirements for an "interactive system" also
specify that the system be easy to operate by someone without any
computer experience, that the the user be prompted for a response, and
that the user's response consist of a single letter or numeral.

Because of these requirements, it was decided to design an interactive system similar to that used in the UCSD Pascal operating system (Ref 12). This interactive system was chosen because the author felt it was the best systm he had used and it meets the requirements of the system. The user is prompted with the allowable responses in the following form: the first letter of the response, a left parenthesis, and the remainder of the word. If one of the possible functions was to setup the system for a new configuration, one of the prompts would be "S(ETUP". If this was the desired function, the user would enter an "S" followed by a carriage return ("<CR>"). Thus. by reading the prompt lines and entering the first letter of his choice, the user can control which function is executed. If an illegal response is made, an error message is printed at the user's terminal and the prompt line is repeated, allowing the user to enter a legal response. A structure chart of the algorithm designed to meet these requirements is shown in Figure 7, and a discussion of the functions of each of the modules which make up the structure chart follows.

MFDM. The Microprocessor Flight Damage Monitor controlling module is the MFDM. This is the module which is executed first when the system is started in the "manual" mode. This module will prompt the user to input the function which he desires to be executed. The user will respond by entering the first letter of the desired function — either S(ETUP, E(XECUTE, O(UTPUT, or Q(UIT. Depending on the input value, one of the subordinate modules SETUP, XECUTE, or CDFOUTPUT will be called, unless the input is a "Q" which will stop program execution.

If an illegal entry is made, the error module will be called to print out an error message and then return control to the beginning of the module. After the called subordinate module completes execution, program control will pass back to MFDM.

SETUP. The functions which the SETUP module must perform are defined by the system requirements. The user must be able to select a specific frequency range and frequency interval, enter the run time for which data is to be processor, and enter a date and a line of comments which are to be printed in the header of the CDF table. In order to make this algorithm easier to modify should future requirements warrant it, this module was designed to be an interactive module. It first prints the last used system configuration and then prompts the user to enter the name of the function he desires to change. Then program control passes to that module. If an illegal entry is made, the error module will be called to print out an error message and then return control to the beginning of the module. By using this method of design, future modifications should only require the adding, deleting, or changing of small special purpose modules to make changes to the



P1g 7. Structure Chart for Configuration Modules

functions performed by SETUP.

GETDATE and GETCOMMENT. These two modules read the data/comment from the terminal as the user inputs it. The date can be up to 13 characters long, and the comments up to 40 characters long. No error checking is performed since whatever is input will be printed in the CDF output table heading in the same format as it was input. If the input is longer than the 13 or 40 characters allowed, the input string will be truncated to the first 13 or 40 characters.

GETFREQUENCY. If the frequency range/interval is to be changed, this module is called. It will first print a list of possible frequency range/interval combinations as shown in Table II. Then it will prompt the user to enter the combination which is desired. As in previous modules, if an illegal response is made the ERROR module will be called with an error message. After control returns from the error module, the list of possible combinations will be printed at the terminal followed by a repeat of the prompt for the user's response.

When a valid response is received, this module will initialize the global variables used in other routines, according to the selected configuration. The variables which must be initialized are: the number of data points to be sampled (N), the frequency interval (INTERVAL), the number of frequency intervals (NFREQ), and the data sampling rate (CLKRATE). After initializing these variables control is returned to the SETUP module where other changes can be requested by the user.

GETTIME. This module procedure will be used if the duration of the test is to be changed. It will accept as input the planned length of the test, in hours, and check that the time is greater than zero.

If an illegal input is received, procedure ERROR will be called with an

appropriate error message followed by a request to enter a new time.

After receiving a valid time, the variable "DURATION" will contain the run time in hours. This completes the discussion of the modules called by SETUP except for the error module which will be discussed next

ERROR. This module will be called by any module which receives an invalid input from the user. The error message to be printed will be passed, as an argument, to this module. The word "ERROR" will be concatenated to the beginning of the error message, and the message will be printed at the user's terminal. After printing the message, program control returns to the calling module.

XECUTE. This module is called when the user wants to start program execution from the terminal. This module will check the run time, and if it is positive a message indicating the start of data processing will be printed at the user's terminal. Program control will then pass to the EXECUTIVE module in the "process data" algorithm. If the run time is less than or equal to zero, the TIME module will be called so that the user can enter the desired run time. After completion of data processing, program control will return to this module. A message indicating termination of data processing will then be printed at the user's terminal, and program control will return to

The last module which can be called by MFDM is the CDFOUTPUT module which prints out the CDF matrix.

CDFOUTPUT. This module prints the CDF matrix at a printer connected to the system. The CDF will be printed in tabular form as a function of frequency and temperature. Each page of the table will

contain a heading which will include the user-supplied date and comments in addition to the time in which data was actually processed.

After printing the CDF matrix, program control will return to MFDM.

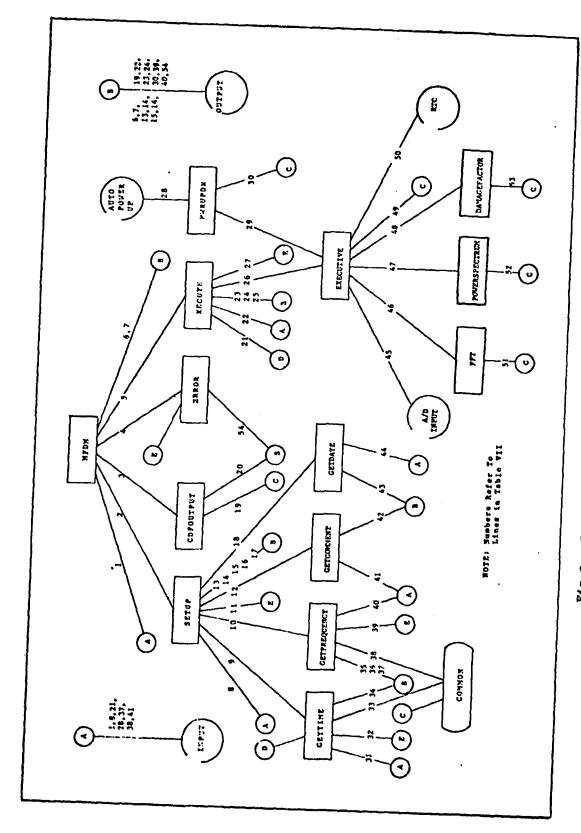
This completes the development of the "configuration" algorithm, except for the definition of the interfaces which will be done when the two algorithms are combined into one system algorithm.

System Algorithm

The system algorithm is the combination of the two algorithms just developed. A structure chart for the system algorithm is shown in Figure 8. The module interfaces are given in Table VII, which is shown in Chapter V. The only requirement that the two separate algorithms do not satisfy is that of the "automatic start-up" mode. The PWRUPDN module fulfills this requirement.

PWRUPDN. This mode of operation requires a non-volatile memory for both the data and the program. When system power is applied, program control will then pass to the beginning of the EXECUTIVE module where the CDF array is cleared. If the power is removed and then reapplied, the power-up sequence will have to check the run time to see if it is still less than the pre-set run time. If so, program control will have to pass to the EXECUTIVE module at a point after the CDF array has been cleared so data processing can continue. If the run time has expired, the program execution will halt.

Having an automatic power-up module implies that something should be done in case of a power loss. This module is designed so that all the power-down module has to do is execute a halt command. Data processing will automatically restart when power is restored.



ţ

Fig 8. Structure Chart for the MFDM System

Summary

This chapter discussed the design of the algorithm which when implemented, will fulfill the requirements established in Chapter II.

The algorithm was designed with specalized modules, each having one function. This design method allows future modifications to be made to a module without having to modify the entire system. The interactive aspect was designed to prompt the user for single letter or numeral responses thus making it easier to operate by someone inexperienced in its operation. The LSI-II which was previously selected as the "best" computer to use for this effort (Ref 7:25) will be presented, with this algorithm in mind, in the next chapter.

IV. Hardware Considerations

Introduction

The algorithm discussed in Chapter III describes the system in detail, but does not discuss its implementation. This chapter presents the decisions that were made concerning the implementation of the system. The decisions made previous to this investigation (Ref 7:23-35) are discussed in the light of the new system requirements.

LSI-11 Microcomputer System

The LSI-IIM Microcomputer manufactured by Norden Systems, Inc. was previously selected as the best computer for this system because of its word size I/O capability, instruction set, memory capability, and availability of software development tools (Ref 7:24).

Norden's LSI-11' is identical, in its functions to the LSI-11 produced by Digital Electronics Corporation (DEC), but uses a different backplane, thus making it incompatible with standard DEC systems. Therefore, a special backplane and card cage would have to be constructed in order to use the Norden LSI-11M with DEC's analog converter and real-time clock that were selected for the MEDM (Ref 7:29-34). Although the Norden LSI-11 is a more durable module (Ref 7:25), DEC's modules are designed to operate in "Minimally Controlled Environmental Conditions" (Ref 6 20) such as the environment in which the MFDM will be operating. Therefore, the LSI-11 was acquired and used for the development of this system. The LSI-11 Microcomputer is capable of handling the additional requirements by using different options than initially determined (Ref 7:25-35).

KD11-J Processor Module. The KD11-F processor module was originally selected for the system, but is not adequate for the real-time processing of data at the higher frequency range of interest (6 K Hz). This is because the KD11-F processor module has a 4 K by 16 bit semiconductor RAM within the module. This memory requires the on-board memory refresh to be enabled which results in an execution stoppage for up to 130 micro-seconds during a memory refresh cycle (Ref 8.76). This delay time, when combined with a 40 micro-second data acquisition service routine, results in an execution delay of up to 170 micro-seconds. This is then the minimum interval between real-time events for which it can be insured that no real-time event will be lost. Thus, the maximum sampling frequency would be 5.8 KHz (1/170 micro-seconds).

The KD11-J processor module does not have the on-board memory so the 130 micro-second delay does not exist because the on-board memory refresh capability is disabled. This reduces the e minimum interval for real-time events to the time required by the data acquisition service routine or the time required for the analog-to-digital converter to digitize a data point, whichever is greater.

The KD11 series processors can all satisfy the power-up auto-start and the manual start as specified by the system requirements (Ref Chap II). These processors have three power-up modes (Ref 8:237). Mode 0, which will be used for this system, allows program execution to start at the address contained in location 24 if the RUN/HALT switch is in the RUN position when power is applied. Otherwise, when power is applied the system is in the halt mode and the user can manually enter the required starting address to start program execution.

Thus, the KD11-J processor module meets the requirements defined in Chapter II, although the system must include a separate memory for the program and data.

MRVII UV-PROM. The requirement that the system be able to start processing data upon application of system power means that the program must be stored in a non-volatile memory such as core memory or programmable read-only memory (PROM). The contents of core memory can accidentally be changed, thus the use of core memory to store the program would not be the best solution. A UV-PROM memory is a read-only memory which makes it difficult to change the program accidentally. The program can be executed directly from a PROM module even though data cannot be stored in the PROM memory. Therefore, the best memory configuration to use is PROM to hold the program, and core memory to hold the data arrays.

A DEC MRV-11 module, which contains up to 4 K words of UV-PROM and 256 words of RAM is available for the LSI-11 (Ref 8:254). Three of these modules (10 K words) will be required to store the complete program. An advantage of UV-PROM is that if modifications to the system are required, the PROM can be re-programmed with the modifications. With the program in PROM, the data arrays and system variables have to be contained in a non-volatile read/write memory such as the MNV-11 Core Memory.

MMVII-A Core Memory. A non-volatile memory is required to store the data array so that it is available to the user after the test run has been performed. The MMV-11 4 K by 16 bit core memory meets the requirements of the system (Ref 7:29). The number of core memory

1.1

maximum size of the CDF array is 1440 elements ((60000 Hz/50Hz)x12 temperature intervals), and the requirement that any two elements in the array can differ by as much as 10 (Ref Chap II) requires that two words (floating point numbers) be used to hold each element in the array (Ref 8:560). This requires 2880 words of core memoy to hold the CDF array. Thus, one 4 K core memory module is adequate.

ADVII-A Analog-to-Digital Converter. The A-to-D converter must be capable of processing analog data at a rate equal to or greater than the Nyquist sampling frequency. For a frequency range of 0-6 K Hz the analog signal must be sampled at a rate of at least 12 K Hz, which is the minimum sampling rate for the analog-to-digital converter.

In order to obtain data at a specific frequency interval, the sampling rate must be controllable so as to satisfy equation 5. The ADVII-A (Analog-to-Digital Converter) and KWVII-A (Programmable Real-Time Clock) are the only modules made for the LSI-II computer which allow the sampling rate to be controlled so as to obtain the required frequency intervals.

The ADVII-A is capable of a 21.5 K Hz sampling rate (Ref 2:23) which exceeds the 12 K Hz sampling rate required for this system. The ADVII-A converter can be controlled by the program or by an external event. In order to control the real-time data acquisition, the conversions can be controlled by the KWVII-A programmable real-time clock.

KWVII-A Programmable Real-Time Clock. This option is available for the LSI-II and can be used to control the analog-to-digital conversions. It can be operated in any one of four programmable modes: single interval, repeated interval, external event timing, and external event timing from zero base (Ref 2:3-1).

This application requires that it be used in mode 1 - repeated interval with the interval being set by the program. The analog-to-digital converter with the programmable real-time clock will handle the data acquisition under program control. The remaining system hardware that is required is the hardware to interface the system to the user.

<u>DLVII-J Serial Line Interface</u>. The DLVII-J is the interface for serial line devices such as the terminal and printer, and the LSI-II bus. There are four independent serial interfaces on the module, and each channel can independently be configured for the following: baud rate (150-38400), number of data bits - 7 or 8, number of stop bits - 1 or 2, and parity - even or odd. Therefore, this module is capable of interfacing with the user's terminal and printer.

DDV11-B Backplane. The DDV11-B is used to hold and electrically connect the different modules. It is used with the HO341 card cage which supports the modules. The backplane will accept the KD11-J processor, core memory, and up to 11 option modules (Ref 8:50). This backplane/card cage combination will hold all of the required modules for the system in one cage, thus making it a portable, self-contained system. The size of the backplane with card cage is 17.1 inches long, 4.8 inches high and 8.7 inches deep (Ref 8:51).

This completes the description of the required LSI-11 hardware.

Digital Equipment Corporation recently introduced two new versions of

the LSI-11 - the LSI-11/2, and LSI-11/23 (Ref 5:55) - either of which could be used for the MFDM system. These two processors are half the width of the LSI-11 but are double the thickness (Ref 5:55-56). As of December 1979, DEC is not yet making a correspondingly smaller analog-to-digital converter, real time clock, or core memory so there is no size gain by using these newer processors. However, the LSI-11/23 is four times faster (Ref 5:57) than the LSI-11 which would increase the cumulative damage matrix update rate if future requirements need a faster rate.

In addition to the microcomputer hardware required, some hardware is needed to acquire the analog signals. This hardware consists of two sensors and associated linear amplifiers which are discussed next.

IBVII-A LSI-11/Instrument Bus Interface. This module is used to allow the computer to control the auto-ranging gain of the vibration signal amplifier. It provides the means of inhibiting the auto-ranging gain, and reading the gain setting of the amplifier before the frequency signal is sampled. Due to a lack of time, this module was never integrated into the bench model system.

KPV11-B Power Fail/Line Time Clock/Terminator. This module controls the power-up and power-down sequences and has provisions for an external clock and a remote console capability (Ref 8:339). With a DC power supply, no line clock will be available. Thus, the external clock input can be used as an input for the system clock. The remote console capability provides the necessary switching controls to allow the system to start up in either the automatic mode or manual mode. Although the module is designed for an AC input, the rectifier circuit

can be bypassed to allow operation with DC power.

Analog Sensors

۱,

In order to gather the required data, two analog sensors are required — one thermocouple and one accelerometer. The AFFDL laboratory supplied the sensors and associated amplifiers for this project. The thermocouple was made from type K thermocouple wire and the accelerometer was a Vibra Metrics VM-112 integrated accelerometer (Ref 13). The accelerometer had a "built in" electronic pre-amplifier allowing its signals to be transmitted over long cables of conventional 2-wire, or coaxial construction, with no loss in sensitivity or spurious cable noise (Ref 13). The output of each sensor goes into an amplifier which amplifies the signal to a level which can be digitized by the analog-to-digital converter.

The amplifiers supplied with the sensors were Intech Model A2583 wide band data amplifiers (Ref 1:1). The gain of the amplifiers can be set either automatically or manually to any one of eight internally programmed gain ranges. The gain setting of the amplifier can be determined by either an analog voltage output or a BCD output, and the auto ranging can be inhibited under program control (Ref 1:1). The thermocouple was connected to one amplifier with the gain set to +60 db. The accelerometer was connected to another amplifier which used a 10 position thumb wheel switch to determine the gain setting. Figure 9 shows the output of the thermocouple amplifier for the temperature range of interest, and Table V shows the gain of the accelerometer amplifier for the 10 possible thumb wheel settings. Table VI shows the specifications for the Intech amplifiers (Ref 1:6).

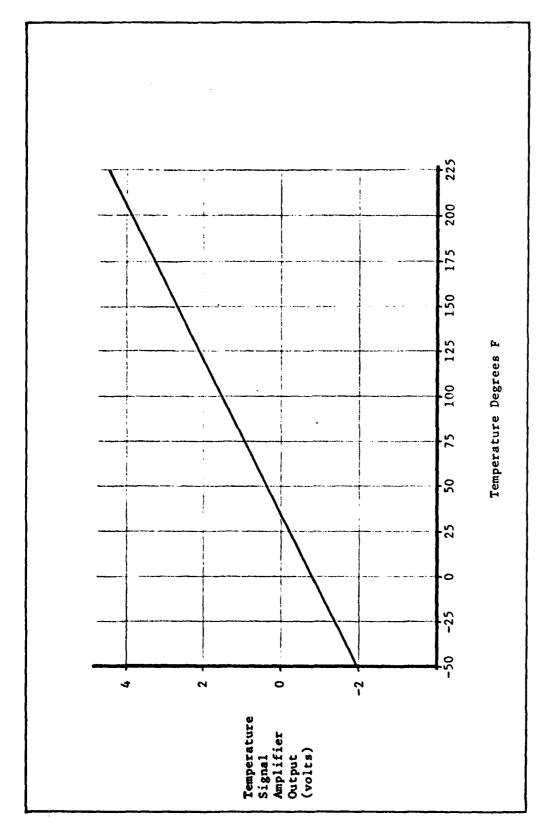


Fig 9. Temperature Signal Versus Temperature

TABLE V

Gain Settings For Frequency Amplifier

Switch Setting	Gain (db)	Digital Gain Code	Analog Gain Code
0	-10	111	-2.40
1	0	110	-1.70
2	10	101	-1.00
3	20	100	-0.35
4	30	011	0.35
5	40	010	1.00
6	50	001	1.70
7	60	000	2.40
8	Auto-		
9	Ranging		

TABLE VI

SPECIFICATIONS

A2583 Automatic Gain Ranging Amplifier

	Ra	nge	}
Functions	llin	Max	Units
Gain Range (10db steps)	-10.0	+60.0	db
Vernier Gain/step	-6 5	+3.5	db/step
Gain Accuracy	0-1	1	db
Gain Stability	0.1		ďЪ
Gain change threshold		Į.	
Down (adjustable)	0.5	2.0	V Riis
UP (adjustable)	10 0	20.0	db below down
Temperature Range	-25 0	+85.0	С
Power Supply Voltage	14.0	16.0	VDC
Current	1	60.0	ma

The design also requires two hardware modules which are not commercially produced for the LSI-11 system. The modules are: a DC power supply and a system clock. The DC power supply will be designed and built by the AFFDL at Wright-Patterson AFB (Ref 11), and is not covered in this report.

System Clock

The system clock must produce a TTL level compatible square wave with a one-second period. This signal will be the external clock input to the KPVII-B module. This module will produce an external interrupt which will drive a software system clock. A clock chip such as the National Semiconductor MM5316 Digital Alarm Clock can be used to generate the 1 second pulse and will also provide a digital readout of the system time if it is desired.

Summary

This section has described the hardware available for the MFDM. The hardware previously selected (Ref 7:23-35) was re-considered in light of the new system requirements. The KD11-F and DLV11 are the only previously selected modules which do not meet the new requirements. The KD11-F processor module has an on-board memory which requires the use of on-board memory refresh and, therefore, could not acquire the data at 12 K Hz as required to obtain the desired frequency range. Thus, the KD11-J processor module was selected. It is the same as the KD11-F processor except there is no on-board memory, and therefore the on-board memory refresh is not required. The DLV11 module is a single port serial interface module. In order to use a terminal and printer, two modules would be required; thus, a DLV11-J

four-port serial interface module was selected. This module can be used for both the printer and the terminal.

Two additional modules, not previously considered, are also required. The first one is an MRVII UV-PROM (Ref 8:137) which is used to store the program, and the second one is an IBVII LSI-11/Instrument Bus Interface (Ref 8:180) which is used to control the auto-ranging gain of the vibration signal amplifier. These additional modules, along with the modules previously selected, establish the hardware requirements for the MFDM. The next chapter uses this information in designing the software to implement the system algorithm.

V. Software Design

Introduction

The next step in the development of the MFDM is to design an algorithm for each of the software modules. In this chapter, the possible program languages available to implement the algorithm are discussed, and a decision is made as to which programming language is the best to use. Each of the required software modules is then developed, insuring that each module will interface properly with the other modules.

Programming Languages

The language in which a module is coded affects the complexity of the program, execution speed, and the ease with which the module can be maintained. The programming languages which were available on the LSI-11 development system were RT-11 assembly language, RT-11 FORTRAN, and UCSD Pascal. The Pascal operating system required 24 K words of memory to operate (Ref 12:iii), and the development system only had 20 K words of memory which could be used without memory refresh when executing the program (Ref Chap IV). Therefore, the Pascal operating system could not be used when executing the program. In addition, no debugging routine was included in the Pascal operating system. Without such a routine, program debugging would be very difficult. Because of these two restrictions, it was decided not to use Pascal to develop the software.

Generally, a structured FORTRAN program is easier to understand than a structured assembly language program which performs the same

task. A FORTRAN program, being easier to understand, will generally be easier to maintain during its life; and therefore, it will have a lower life cycle cost. The disadvantage of a FORTRAN program as compared to an assembly language program is that the execution time is generally longer. Therefore it was decided to code the modules which require the most computational time or which require real-time data processing in assembly language, while the other modules were coded in FORTRAN. It was decided to code the EXECUTIVE and FOURIER modules in assembly language because EXECUTIVE controls the real-time data aquisition and FOURIER, which computes a fast Fourier transform, requires the greatest percentage of execution time (Ref 7 115). The remaining modules were all coded in FORTRAN for ease in future maintenance.

The system algorithm was constructed using top-down structured design, which was considered by the author to be the best design methodology available. The process started with the development of a module which controlled the entire process. This module was then expanded into subordinate modules, each of which performed one major function. These subordinate modules were then expanded into subordinate modules if the major function could be reduced to a series of independent functions. The result of this type of design was a structured system where each module accomplished a specific function, and the coupling between modules was minimized. A structure chart for the complete system is shown in Figure 10, and the module interfaces are shown in Table VII. The following paragraphs describe each of the modules which make up the system in detail, and appendices A and B contain flow charts and the source code for each of the modules.

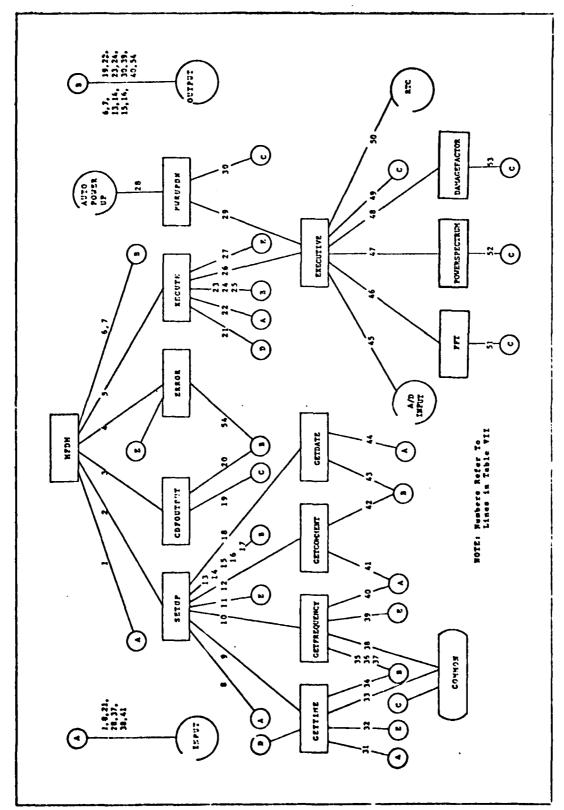


Fig 10. Structure Chart for the MFDM System

TABLE VII

Arguments of the MFDM Modules

	INPUT	OUIPUT
1.	COMMENTS, DATE, DELTAF, DURATION	ANSWER COMMENTS, DATE, DELTAF, DURATION
w 4	CDF, NFREQ, DELTAF, DATE, COMMENTS	
	REENTER LETTER <cr>'</cr>	
i ė	MICROPROCESSOR FLIGHT DAMAGE	MINUTES
	MONITOR.	
	'ENTER S(E1 JP, E(XECUTE, O(UTPUT,	
•	\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\	ANSHED
6		DURATION
10.		DELTAF
11.	THE INPUT MUST BE AN F, R, D, C,	
	Q, OR V'	
12.		COMMENTS
13.	THE DURATION WILL BE', DURATION	
14.	'THE COMMENTS ARE', COMMENTS	
15.	'THE DATE IS', DATE	
16.	'THE FREQUENCY RANGE IS -',	
	NFREQ*DELIAF, 'Hz; DELIAF IS',	
	DELTAF, 'Hz'	
17.	'CHANGE F(REQ, R(UN DURATION, D(ATE,	
	C(OMMENTS, Q(UIT, OR V(ERIFY -	
	? <cr></cr>	
18.		DATE
19.		
20.	CDF OUTPUT TABLE	RTIME

TABLE VII

Arguments of the MFDM Modules (cont.)

	INPUT	OUTPUT
21.		DURATION
22.	Worms alle '21 MOTTA Blin But'	ANSWER
•67	THE DURALLON IS , DURALLON,	
24.	'DATA BEING PROCESSED'	
25.	'DATA PROCESSING COMPLETE'	
26.		
27.	'INPUT MUST BE A Y OR AN N'	-
28.	,	
29.		
30.	PWRFLG	PWRFLG, PTIME, RTIME
31.		DURATION
32.	DURATION MUST BE IN HOURS	
	AND > 0	
33.	MINUTES	
34.	ENTER DIRATION OF TEST (HOIRS)	
	EXAMPLE 15.25 <cr></cr>	
2	57 47 47	
36.	raed range (n nz) Dellar eniek	
	.0 - 6.0 100 1	
	0 - 6.0 50 2	
	0 - 3.0 50 4	
	0 - 3.0 25 5	
	0 - 1.5 100 6	
	- 1.5	
	0 - 1.5 25 8	

TABLE VII

ţ

Arguments of the MFDM Modules (cont.)

OUTPUT	O ENTRY COMMENTS	. DATE ADDATA	ADDATA, B, N, PTIME, RTCMODE, RICRATE, RTIME, TFLG	ADDATA, BITCOUNT, COUNT, INDX, ITVL, INTCELNUM, N, N2, SFACTR ADDATA, ADGAIN, SFACTR PWRSPC,CDFDATA	
INPUT	'ENTER THE NUMBER CORRESPONDING TO THE DESIRED COMBINATION ENTER AN INTEGER BETWEEN 1 AND 7 ? <cr>' B, INDX, INTVL, DELTA 'INPUT MUST BE AN INTEGER BETWEEN 1 AND 7'</cr>		ADDATA, ADGAIN, BITCOUNT, CDFDATA, COUNT, INDEX, INTCELNUM, N, N2, NINTVL, NM, PTIME, PWRSPC, RTIME	_ ,	ERROR', MESSAGE
	37. 38. 39. 40.	43. 44. 45.	4 8	50. 51. 52. 53.	54.

Configuration Modules

The configuration modules are the modules which initialize the parameters used by the data processing modules. These modules are interactive modules which allow the user to configure the system for a particular test. In order to make the configuration process easy for the user to accomplish, the modules were designed to display the system's current configuration and then prompt the user to input the first letter of the function to be executed. Thus, by using prompting messages, the system instructs the user what to do to configure the system. The following sections discuss the design of the modules which are used to configure the system to process data according to the requirements of the user.

MFDM. Program NFDM is the controlling module for the entire system. It is an interactive module which is initally executed when the system is started in the manual mode. It prints the program's name at the user's terminal and then requests that the user enter the first letter of the function to be performed. The format of these two lines is as follows:

"MICROPROCESSOR FLIGHT DAMAGE MONITOR"

"ENTER S(ETUP, E(XECUTE, O(UTPUT, Q(UIT - <CR>"

where <CR> means the user's response followed by a carriage return. If a "Q" is entered by the user, program execution halts. If an "S", "E", or "O" is entered, the appropriate subordinate module is executed. If an illegal entry is made, the system's ERROR module is called with an error message (Table VII, line 4) being passed to it. After the subordinate module completes execution, control is passed back to MFDM.

The modules which are subordinate to MFDM are discussed next.

SETUP. Subroutine SETUP is called when the user wants to configure the system before a test. The module first displays the last used system configuration and asks what is to be changed. If the last configuration was 0-3 K Hz and 25 Hz the following lines would be printed at the user's terminal:

FREQUENCY RANGE:
FREQUENCY INTERVAL:
RUN DURATION:
COMMENTS:
DATE:

0-3000 Hz
25 Hz
(Last Time) Hours
(Last Comments)
(Last Date)

CHANGE F(REQ, R(UN DUR, C(OMMENTS, D(ATE, Q(UIT, or V(ERIFY - ?<CR>

The user's input is then read and, based upon validity, the corresponding module is called. An "F" input calls the GETFREQUENCY module, an "R" calls GETTIME, a "D" calls GETDATE, and a "C" calls GETCOMMENT. The arguments passed to the subordinate modules are shown in Table VII lines 8-18. If a "Q" is entered, program control returns to MFDM; if a "V" is entered, the current system configuration is printed at the terminal. If an invalid input is made, the error module is called - passing the error message shown in Table VII line ll. After the error message is printed, the current system configuration is printed at the terminal, followed by the prompt line requesting another input.

GETFREQUENCY. Subroutine GETFREQUENCY is called by SETUP when the frequency range and/or interval is to be changed. This subroutine prints the possible configuration at the terminal and requests which configuration is desired. This is accomplished by printing the

following lines:

FREQ. RANGE (K Hz)	DELTAF (Hz)	ENTER
0 - 6.0	100	1
0 - 6.0	50	2
0 - 3.0	100	3
0 - 3.0	50	4
0 - 3.0	25	5
0 - 1.5	100	6
0 - 1.5	50	7
0 - 1.5	25	8

ENTER A NUMBER BETWEEN 1 & 8 CORRESPONDING TO THE DESIRED SETUP ?<CR>

An integer is then read from the terminal and used in an arithmetic go-to statement which passes control to an appropriate initialization section corresponding to the desired configuration. If the first character entered is not an integer between 1 and 8, program control falls through the go-to statement, and the error routine is called with the message shown in Table VII line 39. After this, the configuration table is re-printed at the terminal and the user is prompted to enter an integer. When a valid entry is read, control passes to the corresponding label where the following variables are initialized:

COMMON	LOCAL	DEFINITION
В	EXP	The power of 2 which gives the number of data (N) points to be acquired (N=2**EXP) - used in EXECUTIVE.
INDX	INDEX	An integer index used with the sine- look-up table in the subroutine FFT. It is initialized to the offset required for the look-up table.

Intvl	INTERVAL	The number of frequency intervals in the selected frequency range (INTERVAL=MAXFREQ/DELTAF + 1). It is used in the DAMAGEFACTOR and CDFOUTPUT subroutine.
	DELTAP	The frequency interval for which the CDF is to be computed. It is used in the CDFOUTPUT subroutine.
	MAXFREQ	The maximum frequency for which the CDF is to be computed. It is used in the CDFOUTPUT subroutine.
RTCMODE	CLKMODE	CLKMODE is a local variable initialized with a data statement to 11 (base 10). This value sets the real-time clock to 1 MHz (Ref x7:3-12). It is used in EXECUTIVE.
RTCRATE .	CLKRATE	A variable which controls the frequency with which the A/D converter samples the vibration signal. The CLKMODE (initialized to I MHz with a data statement in this routine) divided by the CLKRATE determines the sampling rate. It is used in EXECUTIVE.

After initializing these variables, thus re-configuring the system, program control returns to MFDM.

CETTIME. Subroutine GETTIME is called when the user wants to change the length of time for which the system is to process data. The arguments passed to and from this routine are shown in Table VII line

9. The argument "MINUTES" is a local variable corresponding to the "PTIME" (Planned Time) common block. "PTIME" is the global assembly language variable used in the assembly language routine EXEC. Thus, when the value of "MINUTES" is changed, the global variable is also changed.

This routine first prompts the user to enter the desired run duration, in hours, by printing the following line at the terminal:

"ENTER DURATION OF TEST (HOURS) EXAMPLE 15.25 <CR>

The user then enters the desired run duration and it is stored in the variable "DURATION". This value is tested, and if it is zero or less, the error module is called, passing to it the error message shown in Table VII line 32, and a new time is requested. When a positive time is entered it is multiplied by 60 to form the number of minutes the test will run, which is stored in the global variable PTIME. Program control then goes back to the calling module, returning the new DURATION.

GETDATE. Subroutine GETDATE is called when the user desires to change the test date. The purpose of the test date is to allow the user to have the date of the test printed in the heading of the CDF table.

This subroutine prompts the user to enter the desired date with the following message:

ENTER THE NEW DATE Ex. 14-DEC-79 <CR>

The user then responds by entering the date - which can be up to 13 characters long. The date is stored in the array DATE (13) of type BYTE. No error checking is performed, and if the user enters more than 13 characters, the input is truncated after 13 characters.

GETCOMMENT. Subroutine GETCOMMENT is the same as subroutine

GETDATE except that an array COMMENTS (40) is first cleared before it

is loaded with the user's supplied comments. As with GETDATE, the only

purpose of the COMMENTS is to allow the user to have identifying

comments printed in the heading of the CDF table. The user is prompted

with the following line:

ENTER COMMENTS - 40 CHARACTERS MAX <CR>

and the user then enters up to 40 characters. If more than 40 characters are entered, the input string is truncated to the first 40 characters.

ERROR. Subroutine ERROR is called by MFDM, SETUP, GETFREQUENCY, GETTIME, and XECUTE when an illegal input is entered. The argument passed to this routine consists of an error message up to 40 characters long and in single quotation marks, printed at the user's terminal. ERROR concatenates "ERROR" to the beginning of the message, prints it at the terminal, and returns program control to the calling module. If a message sent to this routine contains more than 40 characters, the message is truncated to the first 40 characters.

CDFOUTPUT. This subroutine is called when the cumulative damage matrix is to be printed. It is designed to operate with a Heathkit H14 printer, and uses its software selectable font size feature to control the size of the print. The table heading is printed at 10 characters per inch, while the table entries are printed at 12 characters per inch. The table is printed as two sections + the first for a temperature range from -50 to 100 degrees Fahrenheit, and the s second for a 75 to 225 degree Fahrenheit range. The 75 and 100 degree temperature intervals were repeated on both sections to make interpolation in this temperature range easier.

In addition, each section consists of up to six pages, depending on the frequency range and interval selected, with 41 lines per page. This allows one page of the table to be printed on an 8 1/2 by 11

inch piece of paper. The table heading is printed on the top of every
page and consists of:

CUMULATIVE DAMAGE MATRIX

HIGHFREO

ŧ.

PAGE (page)

DATE - test date

DURATION - test duration

COMMENTS - user's comments

Following the table heading, one page of the table is printed. An example of the output is shown in Table IX, and is explained in Chapter VII.

The variables used in this routine are defined as follows:

CDF (120,12) The cumulative damage factor array CONTENTS (40) A string for any comments the user wants on the output **DATE (13)** A string to hold the date which will be printed on the output DELTAF The frequency between elements of the CDF array FREQPAGE The frequency range per page of output I,J,K,LGeneral purpose indexes LARGE A three character alpha constant which, when sent to the printer, causes 80 characters per line to be printed LOW, HIGH Two indexing limits used to control the printing of the temperature scale. Their values are: LOW=1=50 deg F; NICH=7=100 deg F LOW=6=75 deg F; HIGH=12=225 deg F

Index limits to control the printing of the frequency in the table LOWTEMP, HIGHTEMP Index limits

Index limits to control the printinh of the temperature in the table

LTSYM

The symbol to the left of the first temperature in the table heading.

It is either a space or a < (less than)

MAXFREQ

The maximum frequency in the CDF array

NFREQ

The number of frequency components

in the CDF array

NPAGE

The number of pages required to print the table for one temperature

range

RMINUTES

The actual duration of the test run

RTSYM

The symbol printed to the left of the last temperature on the table heading. It is either a space or a

> (greater than)

After printing the complete table program control returns to the MFDM module, where, if the user desires another copy of the output table can be obtained by entering another "O".

This completes the discussion of the configuration modules. The next section discusses the execution modules.

Execution Modules

The requirements defined in Chapter II state that there be two methods to initiate data processing. One method required that the user be capable of starting data processing from a terminal, and the second method required that data processing commence with the application of power to the system. The XECUTE module is used to initiate data processing from a terminal. The PURUPDN module is used to initiate data data processing when power is applied if the processor's RUN/HALT

switch is in the run position (Ref Chap IV). An explanation of the two modules follows:

XECUTE. Subroutine XECUTE is an interactive module for usercontrolled data processing. The variables used in this subroutine are
defined below:

Global variable:

PTIME

A label common block with length of one word used to store the planned time (in minutes) for which data processing is to take place.

Local variables:

ANSWER

A variable which contains the user's response. It should contain be either a "Y" or an "N". Anything else

results in an error.

MINUTES

A local variable equal to PTIME

DURATION

The planned processing time in hours.

When called by MFDM, this procedure first checks that PTIME is greater than zero. If it isn't, GETTIME is called to get a new run time. The new run time is then printed at the terminal followed by a prompt asking if it needs to be changed. The user's input is then read, and if it is an "N" (NO), then the following message is printed at the terminal:

"DATA BEING PROCESSED"

and program control passes to the EXECUTIVE module where data processing is initiated. If a "Y" (YES) is entered, then GETTIME is called so a new time can be entered. After the new time is entered,

program control returns to this module and the new time is printed at the terminal. The user is then asked if it needs to be changed. This second printing of the time is to provide a "last chance" recovery capability in case a wrong time is entered. When the user responds with a "Y" or an "N", the process is repeated. If anything other than a "Y" or an "N" is entered, the error module is called with the error message given in Table VII line 27. After data processing is complete, control returns to this module and a "DATA PROCESSING COMPLETE" message is printed at the terminal. Program control then returns to MEDM.

1:

<u>PWRUPDN.</u> Subroutine PWRUPDN controls the start of data processing when the system starts in the "automatic mode". When the processor's RUN/HALT switch is in the "RUN" position and power is applied to the system, this routine is entered via the interrupt vector at location 24.

The global variables/labels which are used in this subroutine are defined below:

GLOBAL VARIABLES/LABELS	DEFINITION
BEGIN	The label of the second instruction in subroutine EXEC. If power fails and comes up in the middle of data processing, control returns to this point.
EXEC	The label of the first instruction in subroutine EXEC. Initial data processing starts at this location.
PTIME	A global variable containing the planned duration of the test.
PWRFLG	A global variable used to indicate if it is the first power-up or not.
PT IME	A global variable used to store the total time the system has been processing data.

This routine first sets location 24 to the address of the power-down sequence (PWRDN) in case of a power failure, and then checks to see if it is the first power-up sequence. If it is, PWRFLG is set equal to one, and program control passes to subroutine EXECUTIVE. If it is not (PWRFLG = 1), then PTIME and RTIME are compared. If RTIME is greater than or equal to PTIME, program execution halts. Otherwise, program control jumps to label "BEGIN" in EXECUTIVE.

After the first power-up sequence, location 24 contains the address of the power-down (PWRDN) sequence. When power fails, the power-down sequence loads location 35 with the address of the power-up sequence (PWRUP) and executes a halt instruction. This allows for multiple power up/down sequences during the data processing period.

Data Processing Modules

The modules discussed in this section - EXECUTIVE, FFT,

POWERSPECRUM, and DAMAGEFACTOR - are the data acquisition and

processing modules. The functions which these modules must perform

are: acquiring and digitizing the analog vibration and temperature

signals, computing the fast Fourier transform of the vibration signal

producing the frequency spectrum which the test item is "seeing",

computing the power in each of the frequency intervals which make up

the frequency spectrum, computing the damage factor for each frequecy

interval, and summing the damage factors into a cumulative damage

matrix. The FFT module was taken from the previous work on this system

(Ref 7:75-85) and modified to meet the additional requirements of this

system (Ref Chap II). Two modules, POWERSPECTRUM and DAMAGEFACTOR, were

written in FORTRAN so as to make future modifications easier - should

they be required - while EXECUTIVE was written in assembly language for speed of execution for the data acquisition. The design of the four modules will now be explained.

EXECUTIVE. The subroutine EXECUTIVE is the controlling module for the data processing. When called from XECUTE, or the first time called from PWRUPDN, a macro-setup routine (INITAL) is called to clear the power spectrum array (PWRSPEC) used in POWERSPEC, clear the CDF array (DATA) used in DAMAGEFACTOR, and to initialize the global variables used in the data processing modules. INITAL uses the global variable B to initialize the following macro-global variables:

BITCOUNT	A counter used in the FFT module's "bit inversion
	routine". It is initialized to half the number of
	binary bits required to write the value of N.

COUNT A counter used in FFT and initialized to half the number of "complex" data points.

INTCELNUM An initial value for the local variable "CELNUM" which is used in FFT's "2nd PASS AND ONWARD" routine.

N The number of real data points to be processed. It is used in EXECUTIVE and FFT and is equal to the value 2N discussed in Chapter III. (A macro variable cannot begin with a number).

N2 The number of "complex" data points. It is equal to N/2 and is used in FFT, POWERSPECTRUM, DAMAGEFACTOR.

NM An index used in FFT's "POST PROCESSING" routine.

It is initialized to the relative location of the N=1st "complex" data point.

PWRFLG A flag set by PWRUPDN upon the first system power up in the auto-start mode. It is initialized to zero.

RTIME A variable used in EXECUTIVE to hold the actual time data has been processed. It is initialized to zero.

TIME A variable used in the EXECUTIVE's time interrupt routine. It is used as a "60's" counter to count 60 seconds. It is initialized to zero.

TSUM A variable used in EXECUTIVE to hold the sum of ten temperature values, which is used to set the global variable TFLG. It is initialized to zero.

In addition to initializing these variables, INITAL also loads the time and power interrupt vectors with their proper values. The next function of EXECUTIVE is to control the acquisition of the temperature data.

The temperature data is acquired using channel 05 of the analog-to-digital converter. This is accomplished by writing 2401 (octal) into the analog-to-digital (A/D) command-status register (ADCSR). This selects channel 05 as the temperature channel and starts the A/D conversion. After the conversion is completed, the 4000 (octal) bias is subtracted, and the converted data is summed into the variable TSUM, which is used to determine the temperature interval in which to store the damage factor. After acquiring the temperature data, EXECUTIVE must control the acquisition of the vibration data. The amplifier used to amplify the vibration signal has an auto-ranging feature which adjusts the gain of the amplifier depending on the magnitude of the vibration signal. Before any vibration data can be digitized, the autoranging feature must be disabled - freezing it at one gain for the sampling period - and then enabling it after the sampling period.

This ability to inhibit the auto-ranging gain has not yet been implemented, due to a lack of time. It will require the use of the IBVII-A LSI-II Instrument Bus Interface module. First, a logical "O" will have to be written to a bit in the instrument bus data register,

which is connected to the inhibit line of the vibration signal amplifier. This will inhibit the auto-ranging feature of the amplifier, allowing one set of data points to be acquired at the same gain setting.

After inhibiting the auto-ranging gain, the BCD gain setting can be read from the instrument bus data register and converted into an integer gain multiplication factor. The multiplication factor is stored as the variable "ADGAIN" which is used in the POWERSPECTRUM module. After acquiring one set of data points, the auto-ranging feature must be released by writing a logical "1" to the inhibit line. This will allow the gain of the amplifier to vary with the intensity of the input signal until the next data set is required.

Since this function has not yet been implemented in EXECUTIVE, the ADGAIN has been set to 1 in EXECUTIVE. After obtaining the multiplication factor, EXECUTIVE loads the ADGSR with 7040 (octal), which selects channel 16 as the channel to be used for the vibration signal, and disables A/D interrupts. The real-time clock (RTC) is then initialized as required by the GETFREQUENCY module and "N" vibration data samples are acquired and stored in the array (ADDATA) under program wait. The reason for using program wait rather than interrupt-driven data acquisition is that there is a 44 micro-second delay after an interrupt is requested before the first interrupt service routine instruction is executed (Ref 8:B-5), during which time nothing is done. This, combined with a 40 micro-second service routine (the length of the routine used in EXECUTIVE), would mean the fastest sampling rate would be 11.9 K Hz which is not fast enough (Ref Chap

III). With program wait, the maximum sampling rate (22.2 K Hz) is determined by the execution time of the service routine (approximately 45 micro-seconds), or the analog-to-digital conversion time (37 micro-seconds), whichever is longer. After the sampling of "N" data points, the real-time clock is stopped and the damage factor is computed.

EXECUTIVE then calls FFT followed by POWERSPECTRUM. The sequence sample data, call FFT, call POWERSPECTRUM - is repeated ten times in order to compute an average power spectrum (Ref Chap III). After this, the value in TSUM is compared to a series of pre-set values which correspond to the temperature intervals. These values, located at label TINTVL, are obtained by taking ten times the theoretical digitized temperature signal for the temperature intervals of interest. After each comparison where TSUM was greater than the value in TINTVL, TFLG is incremented by one. Thus, TFLG will take on a value between 1 and 12 corresponding to the average temperature of the test item. The DAMAGE module is then called to compute the cumulative damage factor and update the CDF array. After this, the variables TFLG, TSUM, and the power spectrum array PWRSPC are cleared, the run time is checked and if RTIME (actual run-time) does not equal PTIME (planned run-time) the procedure is repeated. When RTIME equals PTIME, program control is passed back to the module which called EXECUTIVE either XECUTE or PWRUPDN.

FFT. This module was taken from the previous work performed on a cumulative damage acquisition system (Ref 7:75-85) and modified to allow for different values of "N" as defined in the EXECUTIVE section just discussed. Reference 7 should be consulted to obtain the theory and the actual equations implemented in this routine. This section

will cover the modifications that were made to the module to meet the requirements defined in Chapter II.

Originally, the module was designed to compute a fast Fourier transform (FFT) of 32 data points. In the present system, an FFT of 32,64,128, or 256 data points may be required. To accomplish this, constants which were originally initialization constants were changed to variables initialized by GETFREQUENCY and by INITAL in EXECUTIVE. These global variables are: 1NDX - initialized by GETFREQUENCY; and BITCOUNT, COUNT, INTCELNUM, N, N2, and NM - initialized by INITAL in EXECUTIVE. Each of the variables is defined in the EXECUTIVE or GETFREQUENCY section of this chapter, and Table VIII shows the values to which these variables are initialized.

The largest modification necessary was the modification of the sine-cosine look-up table which is used in the "2nd pass and onward" routine and in the "post processing" routine. The "2nd pass and onward" routine requires the sine and cosine values of (2 * PHI * K)/N2 for K = 0,1,2,...,(N 2/2) - 1, and the "post processing" routine requires the sine and cosine values of (2 * PHI * K)/N for K = 0,1,2,...,(N/2) - 1. One look-up table was constructed containing the sine and cosine values of (PHI * K)/128 for K = 0,1,2,...,127 - which contains all of the required values for N = 32,64,128,256. The appropriate value can then be extracted from the table by adjusting the arguments passed to the look-up routine according to N. This adjustment is accomplished with the global constant INDX. The exact details of how this is done can be found in the source code and in Reference 7.

TABLE VIII

Initialized Constants Used in FFT

N	N2	COUNT	INTCELNUM	NM	INDX	BITCOUNT
256	128	64	32	508	2	7
128	64	32	16	236	4	6
64	32	16	8	124	8	5
32	16	8	4	60	16	4
				L	l	l

This modification, along with the variables initialized by the other modules, allows the FFT module to function with values of N = 32,64,128, or 256. Thus, the FFT module meets the requirements defined in Chapter II in addition to allowing frequency ranges/intervals of 0-10 K Hz at 100 Hz, 0-6 K Hz at 50 Hz, and 0-3 K Hz at 25 Hz (N=256) which, if desired, can be implemented in the future. The output of FFT, the frequency spectrum of the vibrational signal, is stored as complex fixed-point values in the input array ADDATA. The binary point is located between bits 14 and 15, and the scale factor for the array is stored in "SFACTR". After completing the FFT, program control is passed back to EXECUTIVE which then calls POWERSPECTRUM to compute the power spectrum.

POWERSPECTRUM. After the FFT of the input data is calculated, the instantaneous power spectrum for each frequency interval in the FFT spectrum must be computed, except for the zero frequency interval. The power spectrum for the zero frequency interval is not computed because it is undefined (computation would involve division by zero). The computed instantaneous power spectrum is then summed into the array PWRSPEC. The FFT and instantaneous power spectrum calculations are

repeated for ten vibration data sets, thus forming ten times the average power spectrum in the global array PWRSPEC (Ref 7:73.85).

(,

This module was coded in FORTRAN because of the case of coding and future modification. A FORTRAN macro was used to convert the fixed point frequency spectrum values to floating point values. The macro assumes the fixed point value is an integer, converts it to a floating point number, and multiplies it by the scale factor and the amplifier's gain. The macro is encoded as follows:

$$PPDATA(I) = DATA(I) \times 2^{(scale)} \times GAIN$$
, (11)

where DATA(I) is the fixed point value (treated as an integer), SCALE is the scale factor from the FFT module, and GAIN is the gain of the amplifier ADGAIN which is determined in EXECUTIVE. The square of the instantaneous power spectrum is computed by squaring the real and complex floating point values of the frequency spectrum values and adding them together. This value is then summed into the local power spectrum array - PYRSPECTRUM (global array PWRSPC). This is represented mathematically by Eq 12:

POWERSPECTRUM(I) = POWERSPECTRUM(I) +
$$(\text{FPDATA}(K))^2$$
 + $(\text{FPDATA}(K+1))^2$,
I = 1 2,3,...NFREQ, K = 3,5,7,... (12)

where FPDATA(K) and FPDATA(K+1) are the real and imaginery values of the complex frequency spectrum data, and Imax is the maximum number of frequency intervals in the desired frequency range. THE PERSON NAMED IN

The output of this module is the sum of the squares of the instantaneous power spectrums. After ten summations the array PWRSPECTRUM contains ten times the square of the power spectrum - which is the data passed to the damage module.

<u>DAMAGEFACTOR</u>. This module computes the damage factor for each frequency interval, of interest, in the FFT spectrum and sums the values into a cumulative damage matrix. The input arguments to this module are the power spectrum array from POWERSPECTRUM and the temperature flag TFLG from EXECUTIVE. Since the power spectrum array actually contains ten times the square of the average of the power spectrum, Eq 1 is rewritten as:

DAMAGEFACTOR(n) =
$$(K(A'(n))^{3.3})/f(n)^{5.4}$$
 n = 1,2,3,...NFREQ (13)

where
$$\overline{A}'(n) = (POWRESPECTRUM(n))^{0.5}/10$$
, (14)

n ≠ 0 because DAMAGETACTOR(0) is undefined, and NFREQ is the maximum number of frequency intervals in the frequency range of interest. Since K is a constant for all n, and the damage factor is a relative value it can be left out. Re-writing Eq 14 without the "K" term and substituting for A'(n) yields:

$$DF(n) = (PWRSPEC(n)^{1.65})/f(n)^{5.4}, n = 1.2, 3, ..., NFREQ$$
 (15)

where the $10^{3\cdot3}$ term was left out because it is also a constant for all n. Eq 15 was coded as a FORTRAN macro DAMAGE (n.f(n)) and was used to sum the damage factor into the CDF matrix. Using a local variable

TEMP as the global variable TFLG the equation for the CDF matrix is:

$$CDF(n, TEMP) = CDF(n, TEMP) + DF(n)$$
, $n = 1, 2, 3, ..., NFREQ$ (16)

Eqs 15 and 16 were implemented, and the resulting cumulative damage matrix is an nmax by 12 array. The FPT routine is coded for a maximum nmax of 120 (6000 Hz range with a frequency interval of 50 Hz), which means the largest CDF array is 120 by 12, or 1440 elements, which requires 2880 16 bit words. In order to operate with values of N between 32 and 256, the CDF array was defined as CDF (120,12). For values of N less than 256, only part of the array is used.

Summary

A top-down structured design was used to design each of the software modules needed to implement this system. The structure chart. shown in Figure 10, and the module interface table shown in Table VII, describe the system as a whole while appendices A and B contain the flow charts and source code for each of the modules. The next step in implementing the system is the individual testing of the software modules, followed by the assembly and testing of the complete system.

VI. Testing and Experimental Results

Introduction

This chapter discusses both the individual testing of the software modules and the construction and testing of a bench model system. The experimental results obtained from the bench model testing are then discussed, and an explanation of the CDF output table is given.

Testing of Software Modules

Each of the 14 software modules was individually tested for proper operation by writing a controlling program which executed the module being tested, and simulated the other modules. After testing each module, the "configuration" modules were tested as a unit by using a dummy execution routine which simulated the processing of data. This "configuration" system was tested using the RT-11 operating system's On-Line-Debugging-Technique (ODT) (Ref 8:645) to control program execution. Using ODT, the program execution was halted before and after each subroutine call in order to verify the proper operation of the subroutines. After verifying the proper operation of each module in the system, the interactive functions were checked for proper operation and variable initialization. When an error was discovered, the module where the error occurred was checked and corrected.

After completing the tests on the "configuration" system, the "data processing" modules were tested as a unit. First, proper array indexing was verified and then the calculation of the damage factor was checked by a manual calculation. This completed the software testing of the MFDM. The next step was the integration of the hardware and the

software into a complete system.

Integration of the Hardware and the Software

After completing the software testing, the hardware and the software was configured into a complete system which was then tested for functional accuracy.

Bench Model Configuration. In order to assemble a bench model system, the KD11-J processor board had to have the 4 K of on-board memory disabled. This was accomplished by installing wire wrap posts for jumper W9 which must be installed to disable a reply from the resident memory (Ref 8:234). The processor jumpers were then configured as a KD11-J processor (Ref 8:234). The 4 K core memory was set to be the first bank of memory and 16 K of semiconductor memory was set to memory banks one through four. A DDV11-B backplane was not available, so two H9270 backplanes were used - connecting them together with a BCV1B option module (Ref 8:380). Figure 11 shows the system configuration used for the bench model.

Bench Model Testing. To test the computational accuracy of the bench model, it was taken to an AFFDL laboratory where both a shaker table and spectrum analyzing equipment were available. The lab's analog data processing system, which computes the cumulative damage factor, was not working, so a B and K Instrument's graphic spectrum equalizer was used to simulate the vibrational input signal. A Spectral Dynamics Corporation's one-third octave analyzer was connected to the input signal in order to determine the input spectrum. There was no equipment available to test the temperature signal acquisition function, so the temperature input leads were shorted together

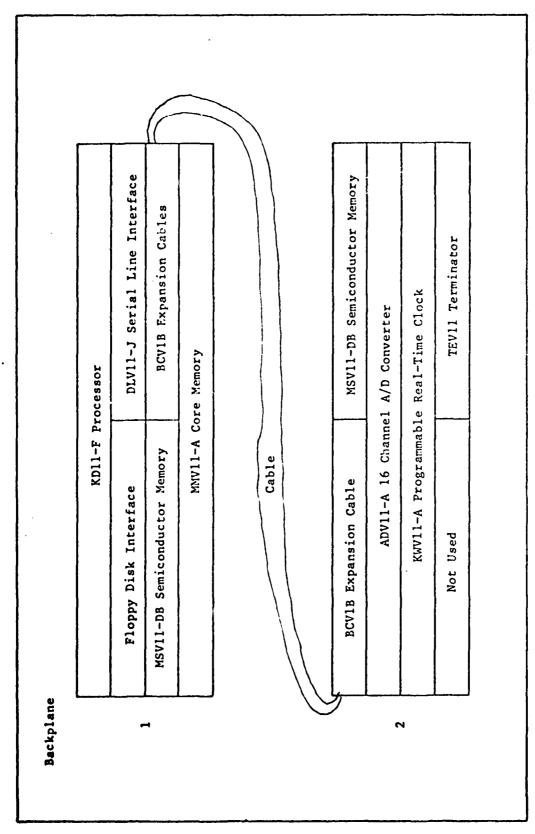


Fig 11. Bench Model Configuration

simulating a O voltage input. After the MFDM processed the simulated data, it was determined that the resolution of the spectrum analyzer was so much less than that of the MFDM that no estimations of the accuracy of the MFDM were possible.

Institute of Technology where the bench model was designed and constructed. The input signals were simulated by using four sine—wave generators to simulate the vibration signal, and a DC power supply was used to simulate the temperature signal. The accuracy of the temperature scale was determined by using a test routine which printed the computed temperature interval corresponding to the DC voltage being used as the simulated temperature input signal. By varying the DC voltage, the voltage at which the computed temperature interval changed was determined. From this voltage, the corresponding temperature at which the computed temperature is changed was computed. This test determined that the temperature scale for the CDF output table is accurate within one degree.

The vibration data was checked by using four sine-wave generators as simulated input signals. Sine wave generators were used in order to have a known frequency spectrum for an input to the system. This allowed the resolution of the system to be determined by varying the frequencies of the input signals until the frequencies were so close as to make them indistinguishable. The smallest difference in frequencies for which the two signals were distinguishable was the resolution.

Using this method, it was determined that the resolution of the CDF output was twice the frequency interval of the output table. The

frequencies are within two frequency intervals of each other, the output table will show a peak at both frequencies. Since the two frequency peaks are next to each other, it is impossible to determine if there was one frequency input between the two intervals, or one frequency input in each interval. By using only one frequency as an input signal, it was determined that the system's accuracy is within 15 Hz when using the 25 Hz frequency interval, within 25 Hz when using the 50 Hz frequency interval, and within 50 Hz when using the 100 Hz frequency interval.

Experimental Results

Without an analog data processing system to compute the damage factor, no accurate method of checking the computational accuracy of the MFDM was available. Also, without an audio frequency spectrum analyzer, no accurate method of checking the output spectrum was available. This left only the testing that was done with the simulated input signals. Table IX shows the results of one of the tests using a system configuration of 0-3 K Hz and an interval of 25 Hz. The input vibration signal consisted of 500 Hz, 835 Hz, 875 Hz, and 2685 Hz sine waves. The amplitudes were 3,5,4, and 1 volt respectively. By varying the input frequency, it was determined that the frequency resolution, for one input signal, is +/- 20Hz up to 6000 Hz. A temperature resolution was not obtained because the thermocouple's linear amplifier did not work.

Explanation of CDF Table. The CDF Table shown in Table IX is an example of the output of the MFDM. The date and comments in the table

TABLE IX

Cumulative Damage Table Output

-		cur	IULATIVE DA	MAGE MATE	:IX	PAGE	1
ATE SURAT:	12-NOU- ION 14.72						
OMMEN			@5V, 875 @	4V, 2685	@1U		
REQ			TEMPERATU			~ =	100
	<-50	-25	Ø	25	50	75	100
25.	0.000E+00	0.0005+00	0.000E+00	0.156E+06	0.000E+00	0.666E+66	0.000E+00
50.	(0.006E+08	0.0008-00	0.000E+00	6.370E+64	0.000E+00	0.000E+00	6.668E+60
75.	0.000E+00	0.0 0005+00	0.0005+00	0.424E+03	0.600E+00	6.666E+66	0.000E+00
100.	0.0005+00	0.0005+00	0.000E+00	0.934E+02	0.000E+00	0.666E+66	0.000E+00
125.	0.0005+00	0.000E+00	0.000E+00	0.290E+02	0.000E+00	0.000E+00	0.000E+00
150.	0.000E+00	0.0005-00	0.000E+00	0.1135+02	0.000E+00	0.000E+00	0.000E+00
175.		0.000E-00	0.000E+00	0.527E+01	0.000E+00	8.000E+09	0.000E+00
	: 0.000E+00	0.000E+00	0.0005+00	0,272E+01	0.000E+00	0.000E+00	0.000E+00
225.		0.000E+00	0.000E+00	0.156E+01	0.000E+00	0.090E+00	0.006E+00
	: 0.000E+00	0.000E+00	0.000E+00	0.963E+00	0.000E+00	0.000E+00	0.000E+00
	0.000E+00	0.000E+00	0.600E+60	0.644E+66	0.000E+00	0.000E+00	0.000E+00
300.	0.000E+00	0.000E+00	0.000E+00	0.451E+00	0.000E+00	0.000E+00	0.000E+00
	0.0005+00	0.0005+00	0.000E+00	0.334E+00	0.000E+00	0.000E+00	0.000E+00
	0.000E+00	0.000E+00	0.000E+00	0.257E+00	0.0005+00	0.000E+00	0.000E+ 00
	0.000E+00	0.000E+00	0.0005+00	0.214E+60	6.000E+00	0.000E+00	0.000E+00
	0.000E+00	0.000E+00	0.000E+00	0.185E+00	0.000E+00	0.000E+00	0.600E+68
	0.666E+66	0.000E+00	0.000E+00	0.179E+00	0.000E+00	0.006E+00	0.000E+00
	(0.000E+00	0.000E+00	0.000E+00	0.204E+09	0.000E+00	0.000E+00	0.000E+00
	: 0.000E+00	0.000E+00	0.000E+00	0.431E+00	0.000E+00	0.000E+00	0.000E+00
	: 0.000E+00	0.000E+00	0.000E+00	0.409E+04	0.000E+00	0.0005+00	0.660E+66
	: 0.000E+00	0.000E+00	0.000E+00	0.257E+00	0.000E+00	0.666E+66	0.000E+00
	4 0.000E+00	8.000E+00	0.000E+00	0.108E+00	0.000E+00	0.000E+00	0.000E+00
_	: 0.000E+00	0.000E+00	0.000E+00	0.948E-01	0.000E+00	0.000E+00	0.000E+00
600.	0.000E+00	0.0005-00	0.000E+00	6.983E-01	0.000E+00	0.000E+00	0.000E+00
	: 0.000E+00	0.0005-00	0.000E+00	0.111E+00	0.000E+00	0.000E+00	0.000E+00
650.	: 0.000E+00	0.000E+00	0.000E+00	0.135E+00	0.666E+66	0.000E+00	0.000E+00
	0.000E+00	0.000E+00	0.000E+00	0.177E+00	0.000E+00	0.000E+00	0.000E+00
700.	1 0.000E+00	0.000E+00	0.000E+00	0.253E+00	0.000E+00	0.000E+00	0.000E+00
	1 0.000E+00	0.0005+00	0.000E+00	0.412E+06	6.000E+00	0.000E+00	0.000E+00
	: 0.000E+00	0.0005+00	0.000E+00	0.804E+00	0.000E+00	0.000E+00	0.000E+00
	1 0.000E+00	0.000E+00	0.000E+00	6.215E+01	0.000E+00	0.000E+00	0.000E+00
	: 0.000E+00	0.000E+00	0.000E+00	0.109E+02	0.000E+00	0.000E+00	0.000E+00
	0.000E+00	0.000E+00	0.000E+00	0.671E+03	0.000E+00 6.000€+00	0.000E+00	0.000E+00
	0.000E+00	0.0002+00	0.000E+00	0.108E+03	6.000E+00	0.000E+00	0.000E+00
875. ***	0.000E+00	0.000E+00	0.0005+00	0.508E+03	0.000E+00	0.000E+00	0.000E+00
	0.000E+00	0.000E+00	0.000E+00	0.701E+00	0.000E+00	0.000E+00	0.000E+00 0.000E+00
925.	0.000E+00	0.0002+00	0.000E+00	0.209E+00	0.000E+00	0.000E+00	
950. 925	0.000E+00	8. 660€+66	0.000E+00	0.814E-01	0.000E+00	0.000E+00	0.000E+00 0.000E+00
975.	(0.000E+00	0.000E+00	0.000E+00	0.374E-01	0.000E+00 0.000E+00	0.000E+00 0.000E+00	0.000E+00
1000. 1025.		0.0005+00 0.0005+00	0.000E+00 0.000E+00	0.193E-01 0.106E-01	0.000E+00	0.000E+00	0.000E+00
	. 0.000.00			~11.00			
	<-50	-25	ø	25	50	75	100
FREQ	(HZ)		TEMPERAT				

TABLE IX

Cumulative Damage Table Output (cont)

		CUM	ULATIVE DA	MAGE MATR	ı×	PAGE	2
ATE	12NOV	-79					
VURAT							
3MMC	NTS 500 HZ	63N 83 5	05U, 875 (14V, 2685 (@1V		
REQ	/U25		TEMPERATU	IRE (DEG.)	E1		
KEG	<-50	-25	0	25	50	75	100
02 5 .	: 0.0005+00	0.0005+00	0,000E+00	0.106E-01	Ú.000E+06	0.000E+00	0.000E+00
	: 0.000E+00	0.0005+00	0.000E+00	0.626E-02	0.000E+00	0.000E+00	0.000E+00
	8.000E+00	0.0005+00	0.000E+00	0.388E-02	0.000E+00	0.000E+00	0.000E+00
	0.0005+00	0.0005-00	0.000E+00	0.249E-02	0.000E+00	0.000E+00	0.000E+00
	1 0.000E+00	0.0005+00	0.000E+00	0.166E-02	6.666E+68	0.000E+00	0.000E+00
	1 0.000E+00	0.000E+00	0.000E+00	0.113E-02	0.000E+00	0.000E+00	0.000E+00
175.	: 0.000E+00	0.0005+00	0.600E+00	0.798E-03	0.000E+09	0.000E+00	0.000E+00
200.	: 0.000E+00	0.0005-00	0.000E+00	0.564E-03	0.000E+00	0.000E+00	0.000E+00
225.	: 0:000E+00	0.0005+00	0.0005+00	0.415E-03	0.000E+00	0.000E+00	0.000E+00
250.	: 0.000E+00	0.000E+00	0.666E+66	0.307E-03	6.000E+00	0.000E+00	0.066E+00
275.	: 0.000E+00	0.0005+00	0.000E+00	0.231E-03	0.000E+00	0.0005+00	0.000E+00
360.	: 0.000E+00	0.000E+00	0.000E+00	0.175E-03	0.0005+00	0.000E+00	0.000E+00
325.	1 0.0005+00	6.0 00€+00	0.000E+00	0.135E-03	0.0005+00	0.000E+00	0.000E+00
350.	(0.000E+60	0.000E-00	0.000E+00	0.105E-03	0.000E+00	0.000E+00	0.006E+06
375.	: 0.000E+00	0.0005+00	0.000E+00	6.831E-04	0.000E+00	0.000E+00	0.000E+00
4 00 .	: 0.000E+00	0.000E+00	0.000E+00	0.671E-04	0.000E+00	0.000E+00	0.000E+00
425.	:: 0.000E+00	0.0005-00	0.0005+00	0.535E-04	0.000E+00	0.000E+00	0.000E+00
450.	: 0.000E+00	0.0002+00	0.000E+00	0.429E-04	0.0005+00	0.000E+00	0.000E+00
475.	0.000E+00	0.000E+00	0.0005+00	0.34KE-04	0.000E+00	0.000E+00	0.000E+00
500.	: 0.000E+00	0.0005+00	0.000E+00	0.297E-04	0.0005+00	0.000E+00	0.000E+00
525.	: 0.000E+00	6.0005+00	0.000E+00	0.251E-02	0.0005+00	0.000E+00	0.000E+00
550.	(0.000E+00	0.000E-00	0.000E+00	0,207E-02	0.600E+60	0.000E+00	6.000E+00
575.	: 0.000E+00	6.000E+00	0.000E+00	0.171E-02	0.000E+00	0.000E+00	0.000E+00
600.	0.000E+00	0.0005+00	0.000E+00	0.144E-02	0.000E-00	0.000E+00	iji)+3iiji), ij
625.	0.000E+00	0.0005+00	0.000E+00	6,122E-02	0.000E+00	0.000E+00	0.000E+00
650.	0.000E+00	0.0005+00	0.000E+60	0.106E-02	0.000E+00	0.000E+00	0.000E+00
675.	0.0005+00	0.0002+00	0. 000E+00	0.101E-02	0.000E+00	0.000E+00	6.666E+66
700.	0.000E+00	0.0005+00	0.000E+00	0.753E-03	0.000E+00	0.000E+00	0.000E+00
725.	: 0.000E+00	0.000E+00	0.000E+06	0.647E-03	0.000E+00	0.000E+00	0.000E+00
750.	(0.000E+00	0.0005+00	0.000E+00	0.6665-03	0.000E+00	0.000E+00	0.000E+00
775.	0.000E+00	0.000E+00	0.000E+00	0.482E-03	0.000E+60	0.000E+00	0.000E+00
800.	1 0.000E+00	0.0005+00	0.000E+00	0.417E-03	0.000E+00	0.000E+00	0.000E+00
825.	0.090E+00	0.0005-00	0.000E+00	0.366E-03	0.0005-00	0.000E+00	0.000E+00
850.	(0.000E+00	0.0005-00	0.000E+00	0.219E-02	0.000E+00	0.000E+00	0.000E+00
875.	: 0.000E+00	0.0 005E+00	0.000E+00	0.284E-03	0.000E+00	0.000E+00	0.000E+00
960.	0.000E+00	0.0005+00	0.000E+00	0.149E-03	6.000E+00	0.000E+00	0.000E+00
925. 980	0.000E+00	0.000E+00	0.000E+00	0.219E-03	0.000E+60	0.0005+00	0.000E+00
950.	0.000E+00	0.0005+00	0.000E+00	0.196E-03	0.0005+00	0.0000+00	0.000E+00
1975.	(0.000E+00	0.0005+00	0.0008+00	0.1765-03	0.000E+00	0.000E+00	0,000E+00
2000.	: 0.000E+00	0.0005+00	0.000E+00	0.155E-03	0.000E+00	0.000E+00	0.000E+00
2025.		* 0.0005+00 0.0005+00	0.000E+00	0.146E-03	0.000E+00	0.000E+00	0.000E+00 0.000E+00
2050.	: 0.000E+00	0.000E+00	0.000E+00	0.126E-03	0.000E+00	0.000E+00	#3+2:00;0;1;0
	/- F 0	_~~	•	~	E O	*76*	100
EDEC	<-50	-25	O TEMPEDOT	25 URE (DEG	50	75	100
スにん	(HZ)		TEMPERAT	una lubu	r /		

TABLE IX

Cumulative Damage Table Output (cont)

ITE IRAT. MMEN			@5U, 875 @	140. 2685 ∙	@1V		
EQ ((HZ) <-50	-25	TEMPERATU 0	JRE (DEG 25	F) 50	75	100
50. ;	0.000E+00.	0.0005+00	0.000E+00	0,1265-03	0.000E+00	0.000E+00	0.000E+0
75. :		0.000E+00	0.000E+00	6.1158-63	0.0005+00	6.666E+66	0.000E+0
ð0. i		0.0005+00	0.000E+00	0.164E-63	0.000E+00	0.006E+00	0.000E+0
25. (0.000E+00	0.000E-00	0.000E+00	0.9585-04	0.000E+00	0.000E+00	0.000E+0
50.		0.000E-00	0.000E+00	0.857E-04	0.000E+00	0.000E+00	0.000E+0
75.		0.0005-00	0.000E+00	0.795E-04	0.000E+00	0.000E+00	0.000E+0
iii.		0.000E+00	0.000E+00	0.737E-04	0.000E+00	0.000E+00	0.000E+0
25.		0.000E+00	0.000E+00	0.697E-04	0.000E+00	0.000E+00	0.000E+0
50.		0.000E-00	0.000E+00	0.6358-04	6.000E+00	0.000E+00	0.000E+0
75.		0.000E+00	0.000E+00	0.602E-04	0.000E+00	0.000E+00	0.000E+6
ÚÜ.		0.000E+00	0.000E+00	0.564E-04	0.000E+00	0.000E+00	0.000E+0
25. (0.000E-00	0.000E+00	0.544E-04	6.000E+00	0.000E+00	0.000E+6
50.		0.0005+00	0.0002+00	0.516E-04	0.000E+00	0.000E+00	0.000E+0
75.		0.000E-00	0.000E+66	0.515E-04	0.000E+00	0.000E+00	0.000E+0
00.		0.000E+00	0.000E+00	0.514E-04	0.000E+00	0.000E+00	0.000E+6
25.		0.000E+00	0.000E+00	0.531E-04	0.000E+00	0.000E+00	0.000E+0
50.		0.0005+00	0.000E+00	0.55JE-04	0.0005+00	0.000E+00	0.000E+0
75.	0.0005+00	0.000E+00	0.000E+00	0.606E-04	0.000E+00	0.000E+00	0.000E+0
90.		0.000E+00	0.000E+00	0.739E-04	6.000E+00	0.000E+00	0.000E+0
25.		0.000E+00	0.000E+00	0.800E-04	0.000E+00	0.000E+00	0.000E+6
50.		0.000E+00	0.000E+00	0.113E-03	0.000E+00	0.000E+00	0.000E+0
	0.0005+00	0.000E+00	6.660E+08	0.173E-03	0.000E+00	0.000E+00	0.000E+0
ÚÚ.		0.0005-00	0.000E+00	0.328E-03	0.000E+00	0.000E+00	0.000E+0
	0.000E+00	0.000E-00	0.000E+00	0.8955-03	0.000E+00	0.000E+00	0.000E+0
50.		0.000E+00	0.000E+60	0.404E-02	0.090E+00	0.000E+00	0.000E+6
	0.000E+00	0.6005-06	0.000E+00	0.414E+00	0.000E+00	0.000E+00	0.000E+0
	0.000E+00	0.0005+00	0.000E+00	0.441E-01	0.000E+00	0.000E+00	0.000E+0
	0.000E+00	0.0005+00	0.000E+00	0.215E-02	0.000E+00	0.000E+00	0.000E+0
	0.000E+00	0.0005+00	0.000E+00	0.487E-03	0.000E+00	0.000E+00	0.000E+0
	0.000E+00	6.000E+00	0.000E+00	0.191E-03	0.000E+00	0.000E-00	0.000E+0
	0.000E+00	0.000E-00	0.000E+00	0.151E-05 0.955E-04	0.000E+00	0.000E+00	0.000E+0
	0.0005+00	0.0005-00	0.000E+00	0.587E-04	0.0005+00	0.000E+00	0.000E+0
	0.000E+00	0.0005+00	0.000E+00	0.399E-04	0.090E+09	8.000E+00	0.000E+0
	0.000E+00	0.0005+00	0.000E+00	0.397E-04	0.000E+00	0.000E+00	0.000E+0
	0.000E+00	0.000E-00	0.000E+00	0.245E-04	0.000E+00	0.000E+00	0.000E+0
	0.000E+00	0.0005+00	0.000E+00	0.189E-04	0.000E+00	0.000E+00	0.000E+0
	0.000E+00	0.000E+00	0.000E+00	0.159E-04	0.000E+00	0.000E+00	0.000E+0
	0.000E+00	0.0005+00	0.000E+00	0.138E-04	0.000E+00	0.000E+00	0.000E+0
	0.000E+00	0.0005+00	0.000E+00	0.120E-04	0.000E+00	0.000E+00	0.000E+0
•		*******		******			
	<-50	-25	0	25	50	75	100

heading are user-supplied and are printed in the same format as input by the user. The duration is the length of time the system actually processed data. If, for some reason, data processing was stopped before the expiration of the planned duration, the duration printed on the table would be the actual time - not the planned time.

Each column of the table corresponds to a 25 degree Fahrenheit temperature range, the center of the range being the temperature printed at the top of the column. If the temperature of the test item is below -37.5 degrees the cumulative damage factor will appear in the column labelled <--50. Also, if the temperature is greater than 217.5 degrees, the cumulative damage factor will appear in the column labelled >225. The temperature range of -50 to +225 is split into two ranges: -50 to +100 and +75 to +225 in order to fit the table on an 8 1/2 x 11" sheet of paper. The 75 and 100 degree columns have been reproduced in both ranges to make analysis of the data at the page boundary easier. Each row of the table contains the CDF at the frequency printed in the left-hand column. After printing 41 lines, the temperature scale is re-printed and a new page is started. Once the complete frequency spectrum is printed for the lower temperature range, the spectrum is printed for the high temperature range, thus completing the CDF Table.

Summary

As stated earlier, due to a lack of time and of the proper test equipment, it was impossible to give the MFDM system a complete testing. The testing with the simulated input signals shows that the system is working properly, although it was not possible to test the

accuracy of its output. The system still needs to be tested with an audio spectrum analyzer, and a test with an analog data processing system should be performed.

VII. Proposed Flight-Worthy System

Introduction

The previous chapters have discussed the design and development of a bench model MFDM. This chapter presents a proposed flight-worthy MFDM system. Proposed hardware will be discussed, including that required to control the auto-ranging gain function of the vibration signal amplifier, which was not implemented in the bench model. The program modifications required to execute the program from PROM are also discussed. In conclusion, the power requirements, as well as a mechanical description of the proposed MFDM system will be presented.

Hardware

The proposed MFDM system configuration, shown in Figure 12, uses a DEC DDV11-B Backplane and an H0341 card cage to hold the hardware modules. The modules are located in the card cage according to their priority - the processor being the highest priority, and the serial interface being the second highest. The position of the remaining modules does not matter since they do not require any specific priority. In addition to holding the required hardware modules, the H0341 has enough space to hold the hardware required for the signal amplifiers and for the system clock. This allows all of the required hardware to be in one chassis, making it easier to package (Ref 11).

The hardware not implemented in the bench model was the controller for the auto-ranging gain of the vibration signal amplifier. The gain must be constant during the data sampling period, and allowed to auto-

	MRV11-BA UV-PROM (OPTIONAL)		Temperature Signal Amplifier	Vibration Signal Amplifier	System Clock Circuit		Sensor Storage Area	
17.125"	KD11-J Processor	DLV11-J Serial Interface	MRV11-BA UV-PROM/RAM	With 4 K PROM	ADV11-A 16 Channel 12 Bit A/D Converter	le Real-Time Clock	mory (4 K Words)	
		MRV11-BA UV-PROM/RAM	With 4 K PROM	Not Used	ADV11-A 16 Channel	KWV11-A Programmable Real-Time Clock	MMV11-A Core Memory (4 K Words)	KPV11-B Power Fail/Line Time Clock/Terminator
		-			4.8"			

Fig 12. Proposed System Configuration

range when data is not being sampled (Ref 11). This allows the intensity of the signals to vary without a loss of data. If the gain of the amplifier is held constant for the duration of the test, then the intensity of some signals may be too small to be digitized, while the intensity of some other signals may be too large for the analog-to-digital converter.

The auto-ranging function adjusts the amplifier gain, within limits, to keep the input signal to the A/D converter less than the 5 volts maximum, thus keeping it within the allowable limits of the A/D converter (Ref 2:2-1). The IBVII-A Instrument Bus Interface Module is designed to control external devices using the "Instrument Bus" (Ref 8:314).

In this case, the module allows the program to control the vibration signal amplifier. In addition, if future uses of the system require the control of multiple signal amplifiers, this module is capable of controlling them (Ref 8:315). In order to control the vibration signal amplifier's auto-ranging gain the inhibit line of the amplifier can be controlled by one bit of the data register, and the 3 bit gain code can be read from the data register (Ref 8:318). The code necessary to use this module will be discussed in the next section.

Software

In order to control the IBV11-A Bus Interface Module, some assembly language code must be added to the EXECUTIVE module. Before the "N" data points are sampled, the auto-ranging feature must be inhibited, and the gain must be stored as the multiplication factor "ADGAIN". This value, which has been set to one in EXECUTIVE until

一 一大大学を大学

the actual code is implemented, is the value by which the data is multiplied in the POWERSPECTRUM module to scale the data.

The valid range for "ADGAIN" was determined by computing the damage factor for the largest and smallest possible values of the power spectrum at the highest and lowest frequency, respectively, and then determining what multiplication factor would cause data overflow or underflow. The largest value for any one element of the power spectrum array is 10⁷ (Ref Chap V), which when used in Eq 15 with a frequency of 25 Hz, gives a damage factor on the order of 10⁷. Since the largest floating point number the computer can store is 10²⁸ (Ref 8:560), the largest multiplication factor ("ADGAIN") is 10²¹ (10²⁸/10⁷).

The smallest "ADGAIN" was determined by using the smallest possible value (1) for an element of the power spectrum array to compute the damage factor at a frequency of 6000 Hz. The damage factor obtained, using Eq 15, was 10^{-21} , and the smallest number which can be repesented by a floating point number is on the order of 10^{-28} (Ref 8:560). Therefore, the smallest "ADGAIN" is 10^{-7} ($10^{-28}/10^{-21}$). If a smaller value is used, the resulting damage factor will be zero due to the 16 bit word size.

After the "N" data points are sampled, the gain must be released to auto-range until the next data sample is required. A flow chart showing the control of the auto-ranging gain of the signal amplifier is shown in figure 13.

In addition to the code necessary to control the amplifier, the program must be stored in MRV11-BA UV-PROM modules. Because a PROM is a read-only memory, the program must be linked so that all variables

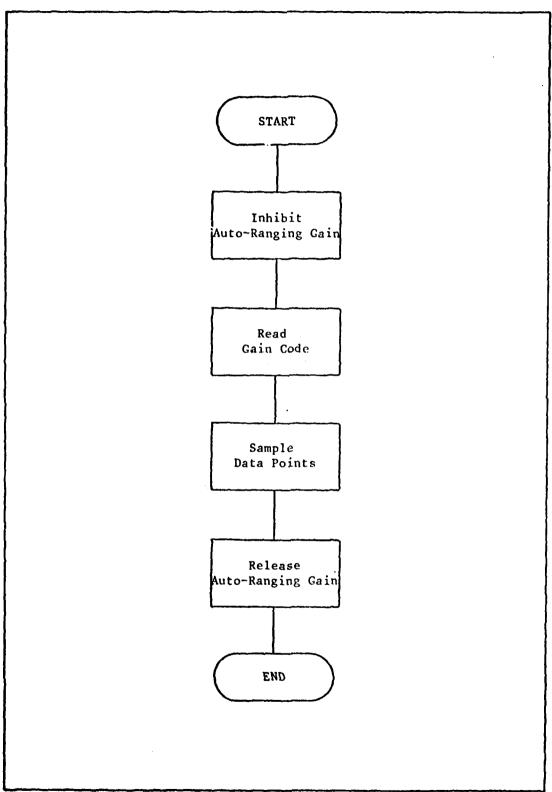


Fig 31. Flow Chart for Auto-Ranging Gain

have addresses in the core memory. Also, there are five instructions which cannot be executed with operands located in PROM. These instructions are MTPS, MUL, DIV, ASH, and ASHC (Ref 8:403). Only two modules use these instructions. EXECUTIVE uses MTPS; and FFT uses MUL, ASH, and ASHC. EXECUTIVE is coded so that it will be executable from PROM, but the FFT module must be modified to store the values for "SINE" and "COS" in the core memory. This can be accomplished by defining the "SINE" and "COS" variables in the PSECT section of the EXECUTIVE module where the CDF array is defined (Ref Appendix B). With this change, all of the modules included in the MFDM system will be executable from the PROM memory.

Figure 12 shows 3 MRV11- BA (4 K by 16 bit PROMS), although the program can be contained in 2 modules - 8 K words. The third PROM is included for future program expansion. Although not a requirement, it is desired to have the CDF matrix plotted in "3 dimensions" (Ref 10). A program to accomplish this can be stored in the third PROM.

Power Requirements

The power requirements for each hardware module are given in Table X (Ref 8:43-46). The total power required is the sum of the values given in Table X plus the power required by the amplifiers and clock. The total power for the modules is 15.4 amps at 5 volts DC and 2.7 amps at 12 volts DC, while the amplifier requires .06 amps at 15 volts DC and the clock requires .003 amps between 8 and 29 volts DC. This power is to be supplied by rechargeable batteries, and they should be capable of operating the system for up to 20 hours (Ref Chap 2).

TABLE X

Proposed Module Specifications (Ref 5:43-46)

Module	Module No.	Description	Power Req +5V	uirements +12V
KD11-J	M7264	LSI-11 Processor	2.0 A	0.9 A
MMV11-A	н223	4 K x 16 Core Memory	7.0 A	0.6 A
DLV11-J	M8043	4-Channel Asynchronous Serial Line Unit Interface	1.3 A	0.2 A
MRV11-BA	M8021	UV-PROM-RAM	0.7 A	0.5 A
KWV11-A	M7952	Programmable Real-Time Clock	1.8 A	0.1 A
ADV11-A	A012	16 Channel 12-bit A/D Converter	2.0 A	0.5 A
IBV11-A	M7954	Instrument Bus Interface	0.1 A	
KPV11-B	M8016YB	Power Fail/Line Time Clock/Terminator	0.6 A	
DDV11-B	н0341	Backplane Card Cage		

AD-A080 417 DESIGN AND IMPLEMENTATION OF A MICROPROCESSOR FLIGHT DAMAGE MON--ETC(U) DEC 79 B C JOHNSON AFIT/GCS/EE/79-6 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 1/3 UNCLASSIFIED 2002 4080417 END 3-80

Packaging

Figure 14 shows the author's concept of the MFDM's final shape, and the layout of the subunits which make up the system is shown in figure 15. Some things which must be considered when designing the package are the cooling and damping of the card cage, the strength of the case and the method by which it will be fastened to the airplane. The two switches on the front panel are keylock switches. The reason for this type of switch is that the "RUN/HALT" switch can be locked in the run position when automatic start-up is required. Then, when the DC power switch is turned on, data processing will start automatically.

The use of keylock switches, although not required, will prevent the inadvertent application or removal of the system's power, or turning the system on with the "RUN/HALT" switch in the "HALT" position when it should be in the "RUN" position. Manual start up can be accomplished by turning the DC power switch on with the "RUN/HALT" switch in the "HALT" position and then switching it to the "RUN" position, or by entering the starting address of the MFDN from a terminal. The package shown in Figures 13 and 14 does not include the power pack which may be a separate box. A separate power supply will reduce the size and allow it to be used with a DC power pack or with an AC power converter pack. The power supply can be included in the system case but this will restrict its use in that only one power source will be available to use.

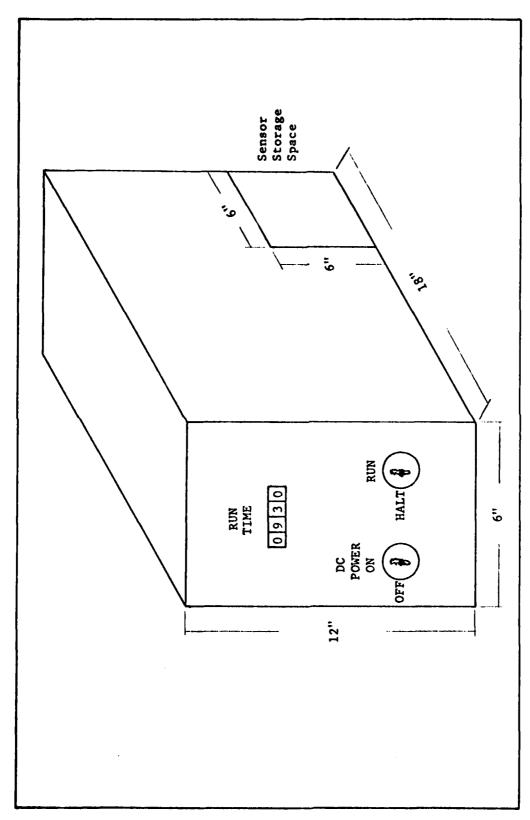


Fig 14. Proposed Concept for the MFDM

The second secon

0

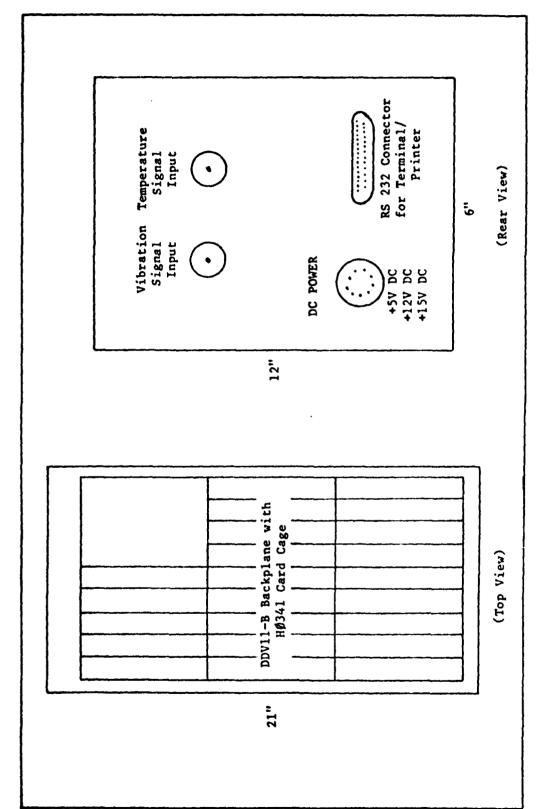


Fig 15. Subunit Layout for Proposed System

1

THE PERSON NAMED IN COLUMN TWO IS NOT THE OWNER.

Summary

In this chapter, the implementation of the flight-worthy system was discussed. The additional hardware needed to control the autoranging feature of the vibration signal amplifier was discussed along with the associated additional code in the EXECUTIVE module required to control the gain of the amplifier. The modifications required to store the program in a UV-PROM were discussed, and the power requirements and packaging of the MFDM were given.

VIII. Recommendations and Conclusions

Introduction

A bench model of the MFDM currently exists. The implementation of a flight-worthy system will require elditional work. This effort would include developing the auto-ranging gain controller, completing the system testing, storing the program in PROM, packaging the system, and constructing the DC power supply required. This chapter presents some recommendations for future development, and some conclusions drawn from this effort.

Recommendations

Recommendations for future work include the installation of the IBVII- A Instrument Bus Interface, and the modification to the EXECUTIVE module necessary to control the auto- ranging amplifier must be completed. The new code should read the amplifier's gain setting from the Instrument Bus Interface and store a multiplication factor equivalent to the gain in the global variable "ADGAIN". The allowable range of "ADGAIN" is from 10^{-6} to 10^{22} , and will insure proper cumulative damage factor computation (Ref Chap VII).

Further testing of the system is required. The testing should be performed using a random vibration signal as an input to the analog sensor, with the output of the sensor going to the MFDM and an audio spectrum analyzer. A vibration table is available at the AFFDL and can be used to generate the input vibration signals for the system and for an audio frequency spectrum analyzer. In addition, an analog data acquisition system can be connected to the same vibration input signals

and can be used to check the computation of the cumulative damage factor. This consists of comparing the peaks in the outputs. The relative difference between the two outputs should be the same for all peaks. The spectrum analyzer can also be used to verify that the peaks in the CDF table are at the same frequency as the input spectrum.

The system program must be stored in UV-PRON. This will allow the system to be run without the disk system that the bench model used. Some minor modifications to the code in the FFT module will be required in order for it to execute from PROM (Ref Chap VII).

The system needs to be assembled in an H0341 card cage with a DDV-11 Back Plane (Ref VII). The clock circuit and signal amplifiers can be included in this card cage, resulting in a self-contained system, except for batteries

The system casing must be constructed to protect the modules from damage in an aircraft environment, and to allow the system to be controlled with the use of two key switches (Ref Chap VII).

A DC power supply must be designed. It will have to supply +5.12, and 15 volts DC at 15.4, 2.7, and .007 amps, respectively, for a period of up to 20 hours of continuous operation (Ref Chap VII). The power supply will connect to the MFDM system via a power line which plugs into the back of the MFDM.

Conclusions

The original design for this system was reviewed with respect to a new set of requirements. The system was then re-designed to meet the new requirements, and a bench model of the MFDM was constructed.

Although time did not allow testing of the complete system, each of the

modules which make up the system was tested individually and found to be accurate within the limits of a 16 bit word size. The complete computer system was tested using sine wave generators to simulate the output of the signal amplifiers. The sine wave generators produced a known input spectrum which allowed the output of the system to be compared to the input signal. The frequency resolution was determined to be half of the frequency interval selected. Due to a lack of time, a complete spectrum analysis comparing the output of the MFDM to a known frequency spectrum input signal was not accomplished.

The current system demonstrates the practicability of using stateof-the-art technology to accomplish the task of real-time data
acquisition and processing. In particular, the real-time data
acquisition of vibration and temperature signals and the real-time
computation of the cumulative damage matrix from these signals was
accomplished. This system thus computes the data required to enable
the AFFDL to produce a damping layer treatment for an item being
tested. Construction of a flight-worthy system can now be accomplished
following the design discussed in Chapter VIII. This investigation has
successfully developed a bench model system for the real-time
processing of the cumulative damage matrix, and has proposed a flightworthy design which can now be constructed.

BIBLIOGRAPHY

- 1. A2583 Instruction Manual. Intech Corp. 1973.
- 2. ADVII-A, KWVII-A, AAVII-A, DRVII User's Manual. Maynard, Massachusetts, Digital Equipment Corporation, 1977.
- 3. Brigham, E. Oran. The Fast Fourier Transform. Englewood Cliffs. New Jersey: Prentice-Hall, Inc., 1974.
- 4. Cooley, J.W., P.W. Lewis and P.D. Welch. "The Fast Fourier Transform Algorithm: Programming Considerations in the Calculation of Sine, Cosine, and Laplace Transform." Journal of Sound and Vibration, 12:315-337 (1970).
- 5. Dulaney, Gerald. "Designing the LSI-11/23." Mini-Micro Systems XII (4):53-60 (April 1979).
- 6. Girmaldi. Frank. Environmental Standards For Computers and Peripherals. DEC STD 102 REV B. Maynard, Massachusetts, Digital Equipment Corporation, 1976.
- 7. Iftekhar, Saleem. Microprocessor Based Data Acquisition and Processing System. Master's Thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology. December 1978. (AD A064728)
- 8. Microcomputer Handbook (second edition). Maynard, Massachusetts, Digital Equipment Corporation, 1977.
- Myers, J. Glenford. Composite/Structured Design. New York: Van Nostrand Reinhold Company, 1978.
- 10. Rogers Lynn. personal interviews. March-November 1979, Aerospace Engineer, Air Force Flight Dynamics Laboratory, Wright-Patterson Air Force Base Ohio.
- 11. Talmadge, R. D., personal interviews June-November 1979, Aerospace Engineer, Air Force Flight Dynamics Laboratory, Wright-Patterson Air Force Base, Ohio.
- UCSD (Mini-Micro Computer) Pascal. Version II.0 Institute for Information Systems, UCSD Mailcode C-021, La Jolla, California, March 1979.
- 13. VM-112 Integrated Accelerometer Short Form Brochure. Vibra Metrics Inc., October 1974.

Appendix A

Flow Charts and Source Code for Configuration Modules

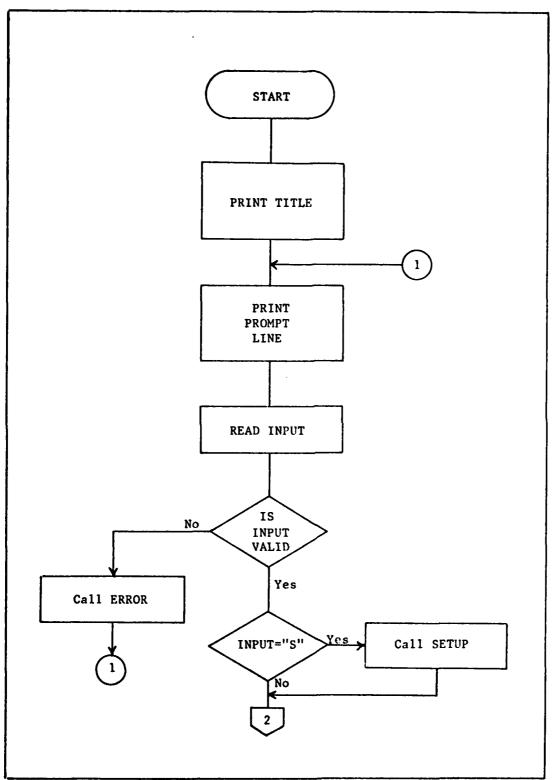


Fig 16. Flow Chart for MFDM

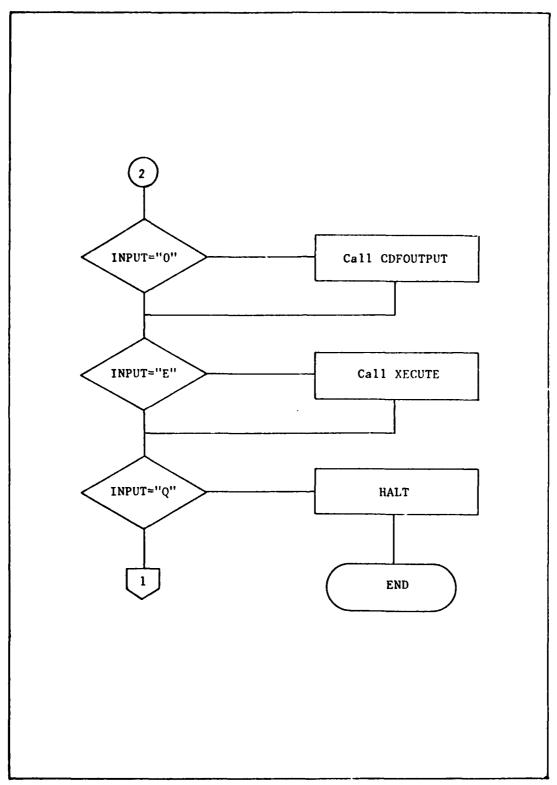


Fig 16. Flow Chart for MFDM (cont)

```
FOPTRAN IV
           V02.1-11
                      Wed 28-Nov-79 00:00 00
                                                             PAGE 001
0001
            PROGRAM MFDM
     C
            THIS MICROPROCESSOR FLIGHT DAMAGE MONITOR (MFDM)
     C PROGRAM IS AN INTERACTIVE PROGRAM WHICH DOES THREE THINGS
     C ALLOWS THE USER TO CONFIGURE THE SYSTEM AS DESIRED. ALLOWS
     C THE ESER TO MANUALLY CONTROL DATA PROCESSING, ALLOWS THE USER
     C REQUEST AN OUTPUT OF THE CDF MATRIX.
            THIS PROGRAM IS STARTED BY TYPING RU MEDM IF UNDER RT-11.
     C OR BY TYPING 1000 G IF UNDER SYSTEM ODT. DATA PROCESSING
     C CAN BE STARTED MANUALLY BY TURNING THE SYSTEM HALT/RUN TO RUN
     C AND THEN TURNING THE DC POWER SWITCH ON.
            WHEN THE PROGRAM IS STARTED EITHER OF THE FIRST TWO WAYS.
     C THE USER WILL BE GREETED BY:
     С
           MICROPROCESSOR FLIGHT DAMAGE MONITOR
     С
           ENTER S(ETUP, E(XECUTE, O(UTPUT, Q(UIT
                                                      ?<CR>
     C UHEN THE USER ENTERS THE FIRST LETTER OF THE SUBROUTINE DESIRED
     C FOLLOWED BY A CARRIAGE RETURN (<CR>), THAT SUBROUTINE WILL BE
     C EXECUTED.
     C
           SUBROUTINE SETUP INITIALIZES THE VARIABLES USED IN THE
     C DIFFERENT SUBROUTINES TO THEIR INITIAL VALUES. THE DEFAULT
     C VALUES ARE THE LAST USED VALUES, WITH THE VALUES INITIALIZED TO
     C A FREQUENCY PANCE OF 0-3000 HZ UITH A FREQUENCY INTERVAL OF
     C 100 HZ THE FIRST TIME THE PROGRAM IS RUN.
           SUBROUTINE XECUTE CONTROLS THE DATA ACQUISITION IN
     C A CONTROLLED ENVIRONMENT SUCH AS A LABORATORY WHERE A TERMINAL
     C IS AVAILABLE.
           SUBROUTINE COFOUTPUT PRINTS OUT A CUMULATIVE DAMAGE MATRIX
     C FOR FREQUENCY VS TEMPERATURE.
           FORTRAN CO'MON BLOCKS (USED TO TRANSFER INFORMATION TO
     C AND FROM THE ASSEMBLY SUBROUTINES).
     C
     C
            ANSUER
                         !USED TO HOLD THE ANSWER TO YES/NO QUESTIONS
     C
            CDF (1452)
                         !THE CUMULATIVE DAMAGE PACTOR ARRAY
            COMMENTS (40) IAN ARRAY OF ALPHANUMERICS WHICH IS PRINTED
     C
                         ION THE OUTPUT TABLE.
            DATE(13)
                         IAN ARRAY TO HOLD THE DATE
            DELTAF
                         ITHE SIZE OF THE FREO INTERVAL IN THE
                         !CDF ARRAY
            DURATION
                         ITHE DURATION OF A TEST IN HOURS
```

IAND FOURIE

ITHE EXPONENT FOR N=2**EXP WHERE

1THE DURATION OF A TEST IN MINUTES

!EXP IS THE LOCAL VARIABLE CORRESPONDING

!TO THE GLOBAL VARIABLE B USED IN EXEC

EXP

MINUTES

C

C

```
ITHE NUMBER OF FREQ INTERVALS IN THE
      C
             NFREQ
      C
                           !CDF ARRAY
      C
                         · !A FLAC USED IN THE POWER-UP ROUTINE TO
             POWERFLAG
                           !INDICATE THE FIRST "AUTOMATIC" POWER-UP.
      C
         VARIABLE TYPE DEFINITION
      C
0002
            LOGICAL*1 ANSWER, DATE(13) COMMENTS(40) POWERFLAG
0003
            INTEGER*2 EXP, NFREQ
0004
            INTEGER*2 MINUTES, INDEX
0005
            REAL*4 DURATION, DELTAF, CDF (120, 12)
        GLOBAL VARIABLES DEFINED
            COMMON /CDFDATA/CDF,/PTIME/MIMUTES,/DELTA/DELTAF
0006
0007
            COMMON /B/EXP,/INDX/INDEX /NINTVL/NFREQ,/PWRFLG/POWERFLAG
      C
      C
        SUSTEM DEFAULT ARGUMENTS INITIALIZED
0008
            DATA EXP INDEX, NFREQ DELTAF/6.8, 30, 100-2/
            DATA DATE/'1' 4','-','D' E','C','-','7','9',4*' '/
0009
0019
            DATA MINUTES/30/
0011
            DATA POWERFLAG/0/
                                   1POWERFLAG IS INITIALIZED TO 1
                                   IFOR USE IN THE POWER-UP ROUTINE.
      C
      C
        PROGRAM EXECUTION STARTS HERE. TITLE AND A REQUEST FOR THE USERS
        RESPONSE IS PRINTED AT THE TERMINAL FIRST.
0012
            TYPE 100
0013
       100 FORMAT (2X, MICROPROCESSOR FLIGHT DAMAGE MONITOR')
0014
       200 TYPE 300
0015
       300 FORMAT (/2X. ENTER S(ETUP, E(XECUTE, O(UTPUT, Q(UIT - ?<CR>'$)
0016
            ACCEPT 400, ANSWER
0017
       400 FORMAT (A1)
        THE USERS ANSUER IS CHECKED AND IF IT IS VALID PROGRAM CONTROL
      C PASSES TO THE REQUESTED ROUTINE.
0018
            IF ((ANSWER.NE. S').AND.(ANSWER.NE.'E').AND.(ANSWER.NE.'O').AND.
           1(ANSWER.NE.'Q')) CALL ERROR ('INPUT MUST BE AN S E.O.OR Q')
      C
0020
            IF (ANSUER.EQ. S')
           1CALL SETUP (DELTAF, NFREQ, DATE, COMMENTS, MINUTES/60.)
      C
0022
            IF (ANSWER-EQ-'E') CALL XECUTE
0024
            IF (ANSWER-EQ 'O')
           1CALL COFOUTPUT (CDF NFREQ DELTAF, DATE, COMMENTS)
      C
0026
            IF (ANSWER-EQ. Q') STOP 'ALL DONE'
0028
            GO TO 200
0029
            END
```

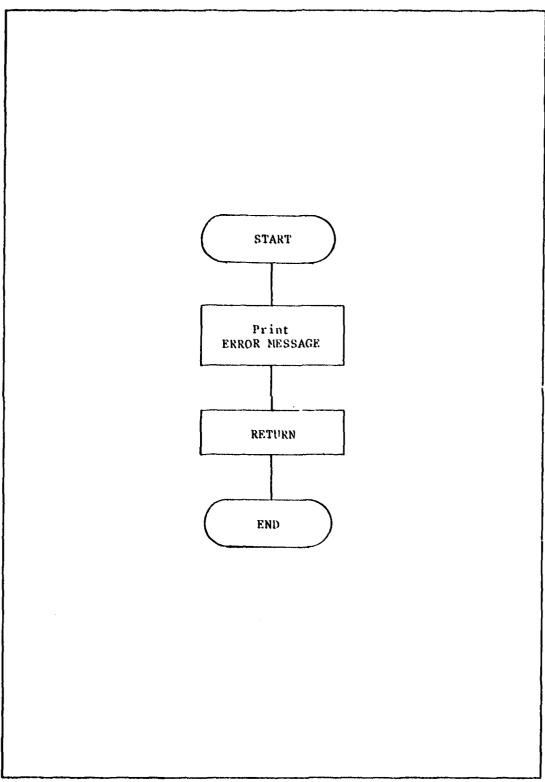
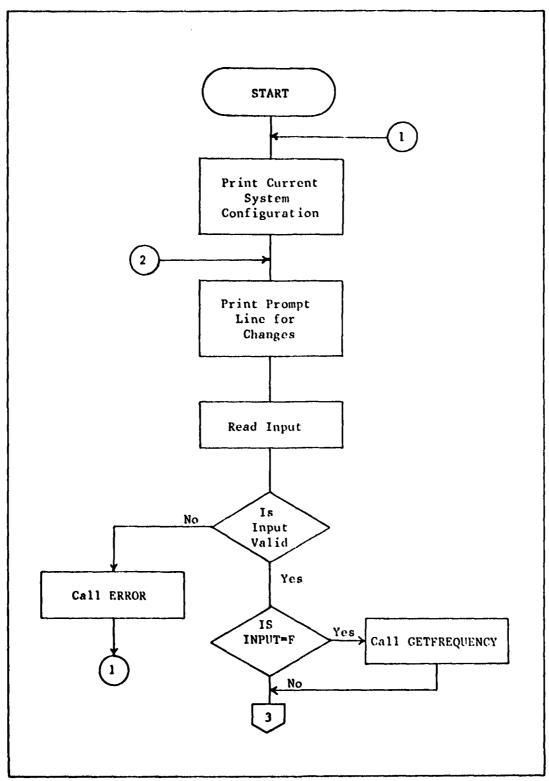


Fig 17. Flow Chart for ERROR

```
PAGE 001
FORTRAN IV VO2.1-11 Fri 23-Nov-79 14:55:18
0001
           SUBROUTINE ERROR (MESSAGE)
     C
           ERROR ROUTINE WHICH TYPES THE ERROR MESSAGE "MESSAGE".
     C
     C "MESSAGE" IS A STRING OF TYPE LOGICAL*1 OF LENGTH UP TO 40.
0002
           LOGICAL*1 MESSAGE(40)
0003
           TYPE 100, MESSAGE
     100 FORMAT (/2X 'ERROR - ', 40A1)
0004
0005
           RETURN
0006
           END
```



.....

Fig 18. Flow Chart for SETUP

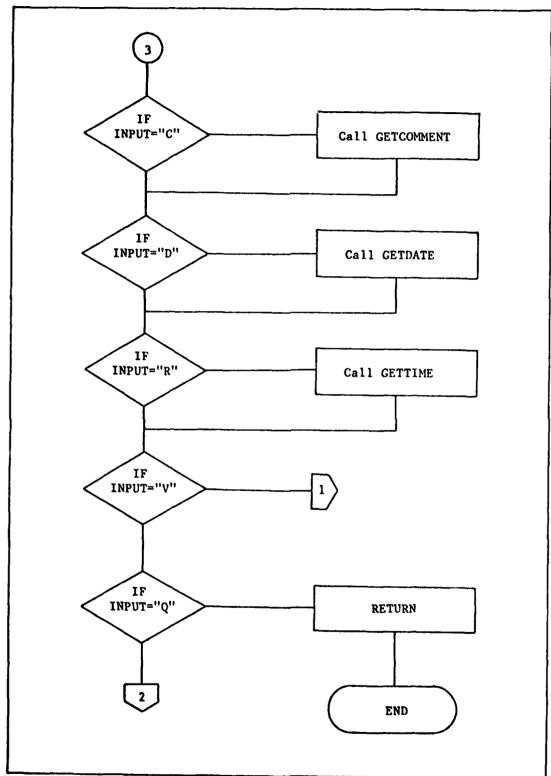


Fig 18. Flow Chart for SETUP (cont)

FORTRAN IV VO2-1-11 Tue 27-Nov-79 00:00:00 PAGE 001 0001 SUBROUTINE SETUP (DELTAF, NFREQ, DATE, COMMENTS, DURATION) SETUP CONTROLS THE INITIALIZATION OF THE PROGRAM VARIABLES. C IT IS AN INTERACTIVE ROUTINE BY WHICH THE USER CAN CHANGE THE C FREQUENCY RANGE AND INTERVAL FOR WHICH DATA IS TO BE PROCESSED. C IN ADDITION THE USER CAN CHANGE THE RUN DURATION, DATE AND COMMENTS. C C THREE SUBROUTINES ARE CALLED BY THIS SUBROUTINE. THEY ARE: C GETFREQUENCY(DELTAF) - A ROUTINE WHICH INITIALIZES THE C GLOBAL VARIABLES USED IN THE ASSEMBLY C LANGUAGE ROUTINE EXEC. ALSO VARIABLES MAXFREQ AND DELTAF ARE SET TO THE MAXIMUM FREQUENCY IN THE CDF ARRAY, C AND DELTAF IS THE FREQUENCY INTERVAL C IN THE CDF ARRAY. C GETTIME (DURATION) - A ROUTINE WHICH READS THE DESIRED RUN C TIME FROM THE TERMINAL AND STORES IT IN C C DURATION (IN HOURS) , AND IN MINUTES (IN MINUTES). C C C VARIABLES USED IN THESE ROUTINE ARE C C ANSWER - LOCATION WHERE THE RESULTS OF A Y/N QUESTION C IS STORED. COMMENTS (40) - A 40 BYTE STRING TO CONTAIN ANY MESSAGE THE USER WANTS TO BE PUT IN THE OUTPUT TABLE. C - A 9 BYTE STRING TO HOLD THE DATE WHICH IS DATE (13) C PRINTED ON THE OUTPUT TABLE C DELTAF - THE FREQUEUCY INTERVAL IN THE CDF ARRAY. C DURATION - THE PLANNED RUN TIME IN HOURS C - THE NUMBER OF FREQUENCY INTERVALS IN THE CDF NFREQ C ARRAY. 0002 LOGICAL*1 ANSWER, DATE (13) COMMENTS (40) 0003 INTEGER*2 NFREQ

REAL*4 DURATION, DELTAF

0004

```
C
            THIS ROUTINE, BEING INTERACTIVE. FIRST DISPLAYS THE CURRENT
      C VALUE OF A VARIABLE AND THEN ASKS IF IT SHOULD BE CHANGED.
        USER RESPONDS WITH A YES (Y) OR NO (N). IF IT IS TO BE CHANGED
        THE USER IS PROMPTED TO ENTER THE NEW VALUE.
            FOR THE DATE AND COMMENTS NO ERROR CHECKING IS DONE SO
      C
      C WHAT EVER IS ENTERED (PRINTABLE CHARACTERS ONLY) WILL BE PRINTED
      C ON THE OUTPUT TABLE.
      C
0005
       10
            TYPE 20, NFREQ*DELTAF. DELTAF
            FORMAT(///2X, 'FREQUENCY RANGE: 0 - ' F6-1.' HZ'/
0006
       20
                                       'F5.1,' HZ')
           12X 'DELTAF:
0007
            TYPE 30, DURATION
                                           ',F5.2,' HOURS')
8000
       30
            FORMAT (2X 'RUN DURATION:
0009
            TYPE 40 COMMENTS
0010
       40
            FORMAT (2X, 'COMMENTS: ', 10X. 40A1)
0011
            TYPE 50, DATE
0012
       50
            FORMAT(2X, 'DATE: ', 14X, 13A1)
0013
       60
            TYPE 70
0014
            FORMAT (///2X, 'CHANGE F(REQ, R(UN DUR C(OMMENTS',
           1' D(ATE, Q(UIT, OR V(ERIFY - ?<CR>'$)
0015
            ACCEPT 80, ANSUER
0016
       80
            FORMAT(A1)
      C
0017
            IF ((ANSWER.NE.'F').AND.(ANSWER.NE.'R').AND.(ANSWER.NE.'C').AND.
           1(ANSWER.NE.'D').AND.(ANSWER.NE.'Q').AND.(ANSWER.NE.'V')) CALL
           2 ERROR(' INPUT MUST BE A: F,R C D Q,OR V')
      C
            IF (ANSWER .EQ. 'F') CALL GETFREQUENCY(DELTAF)
IF (ANSWER .EQ. 'R') CALL GETTIME(DURATION)
0019
0021
            IF (ANSWER .EQ. 'C') CALL GETCOMMENT (COMMENTS)
0023
            IF (ANSWER .EQ. 'D') CALL GETDATE (DATE)
0025
            IF (ANSWER .EQ. 'V') GO TO 10
0027
0029
            IF (ANSWER .EQ. 'Q') RETURN
0031
            GO TO 10
                               ICHANGED FROM 60 TO 10 AFTER THESIS WAS
                              PRINTED - VERIFY IS NOW AUTOMATIC.
0032
            END
```

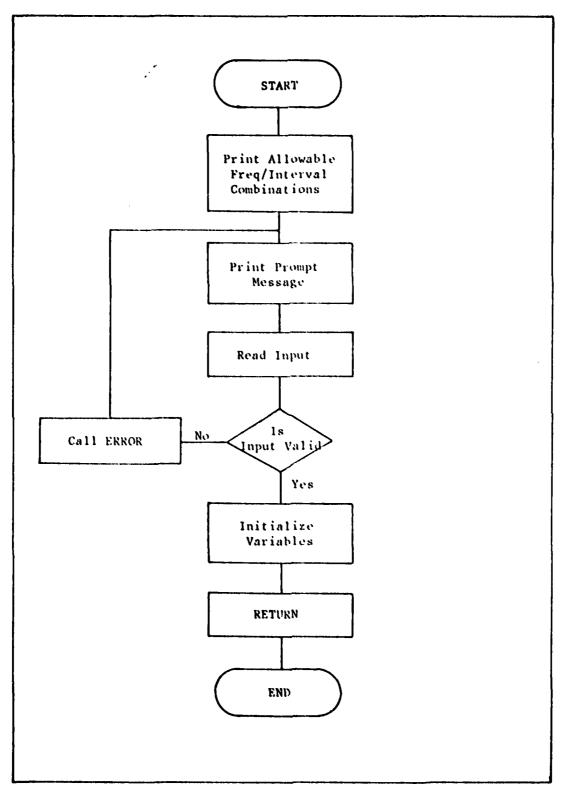


Fig 19. Flow Chart for GETFREQUENCY

```
Fri 23-Nov-79 14:54:13
FORTRAN IV
             V02.1-11
                                                                PACE 001
0001
            SUBROUTINE GETFREQUENCY
      C
      C
            CHANGEF IS USED TO INITIALIZE THE FOLLOWING VARIABLES
      C
      C
          COMMON
                      LOCAL
                              DEFINITION
      C
      C
          /B/
                         EXP
                             - THE INTEGER WHICH SATISFIES
      C
                              THE EQ. N=2**EXP. WHERE "N"
      C
                              IS THE NUMBER OF COMPLEX"
      C
                              DATA POINTS PROCESSED.
      C
      C
         /DELTA/
                             - THE FREQUENCY INTERVAL IN
                      DELTAF
      C
                              THE CDF ARRAY.
      C
      C
          /INDX/
                      INDEX
                              - AN INTEGER INDEX USED WITH
      C
                              THE SINE LOOK-UP TABLE IN THE
      C
                              FFT ROUTINE.
      C
      C
          /NINTVL/
                              - THE NUMBER OF FREQUENCY INTERVALS
                      NFREQ
      C
                              IN THE CDF ARRAY.
      C
      C
          /RTCMODE/
                     CLKMODE - THE MODE OF OPERATION OF THE
      C
                              PROGRAMMABLE REAL TIME CLOCK.
      C
                              POSSIBLE VALUES ARE:
      C
      C
                               CLKNODE
                                               FREQUENCY
      C
                               (DECMIAL)
      C
      C
                                11
                                                     MHZ
                                                 1
      C
                                19
                                               100
                                                     KHZ
      C
                                27
                                                     KHZ
                                                10
      C
      C
      C
           /RICRATE/ CLKRATE - THE FREQUENCY DIVISOR FOR THE CLKHODE
      C
      C
      C
      C
            VARIABLE TYPE DEFINITION
      C
0002
            INTEGER*2 EXP INDEX. INTERVAL, ENTRY NEREQ
0003
            INTEGER*2 CLKHODE CLYRATE
0004
            REAL*4 DELTAF
      C
         GLOBAL VARIABLES
      C
0005
            COMMON /B/EXP./INDX/INDEX./NINTVL/NFREQ./DELTA/DELTAF
0006
            COMMON /RTCMODE/CLKMODE /RTCRATE/CLKRATE
      C
      C
        DEFAULT REAL-TIME CLOCK SETTINGS
      C
0007
            DATA CLKNODE /11/.CLKRATE /-156/
```

```
C
      C
      C
            THE POSSIBLE COMBINATIONS OF THE FREQUENCY RANGE AND FREQUENCY
      C INTERVAL ARE DISPLAYED FIRST. THE USER IS THEN PROMPTED TO ENTER
      C THE INTEGER (1-6) CORRESPONDING TO THE COMBINATION DESIRED. THE
      C INTEGER IS USED IN A COMPUTED GO TO STATEMENT TO BRANCH TO THE
      C APPROPRIATE LOCATOIN FOR VARIABLE INITIALIZATION.
      C
      C
8000
       100 TYPE 200
0009
       200 FORMAT (//2X, 'FREQ. RANGE (KHZ)
                                              DELTAF (HZ)',9X,'ENTER'/)
0010
            TYPE 300, 6., 100.2, 1, 6., 50.1, 2, 3., 100.2, 3, 3. 50.1, 4 3., 25.0, 5,
           91.5, 100.2, 6, 1.5, 50.1, 7, 1.5, 25.0, 8
0011
       300 FORMAT(8(8X F4.1,12X F5.1,15X.11/))
0012
            TYPE 400
0013
       400 FORMAT (//1X. 'ENTER A NUMBER BETWEEN 1 & 8 CORRESPONDING TO THE',
           1' DESIRED SETUP ?<CR>'$)
0014
            ACCEPT 500, ENTRY
       500 FORMAT(II)
0015
0016
            GO TO (1 2,3,4,5 6,7.8) ENTRY
                                               !IF ENRTY>8 OR <1 CONTROL
      C
                                               !FALLS THROUGH.
0017
            CALL ERROR ('INPUT MUST BE AN INTEGER BETWEEN 1 AND 8')
0018
            GO TO 100
      C
        VARIABLE INITIALIZATION STARTS HERE
      C
0019
         1 EXP=7
0020
            INDEX=4
0021
            NFREQ=60
0022
            DELTAF=100.2
0023
            CLKRATE=-78
            GO TO 600
0024
0025
         2 EXP=S
0026
            INDEX=2
0027
            NFREQ=120
0028
            DELTAF=50.1
0029
            CLKRATE=-78
0030
            GO TO 600
0031
         3 EXP=6
0032
            INDEX=8
0033
            NFREQ=30
0034
            DELTAF=100.2
0035
            CLKRATE=-156
            GO TO 600
0036
         4 EXP=7
0037
            INDEX=4
0038
0039
            NFREQ=60
0040
            DELTAF=50.1
0041
            CLKRATE=-156
0042
            GO TO 600
```

· 中の中の日の日の

```
0043
         5 EXP-8
0044
            INDEX-2
0045
            NFREQ-120
0046
            DELTAF=25.
0047
            CLKRATE =-156
0048
            GO TO 600
0049
         6 EXP=5
0050
            INDEX=15
0051
            NFREQ=15
0052
            DELTAF=100.2
0053
            CLKRATE=-312
0054
            GO TO 600
0055
         7 EXP=6
0056
            INDEX=8
0057
            NFREQ=30
0058
            DELTAF=50.1
0059
            CLIRATE=-312
0060
            GO TO 600
0061
         8 EXP=7
0062
            INDEX=4
0063
            NFREQ=60
0064
            DELTAF=25.
0065
            CLKRATE=-312
0066
       600 RETURN
0067
            END
```

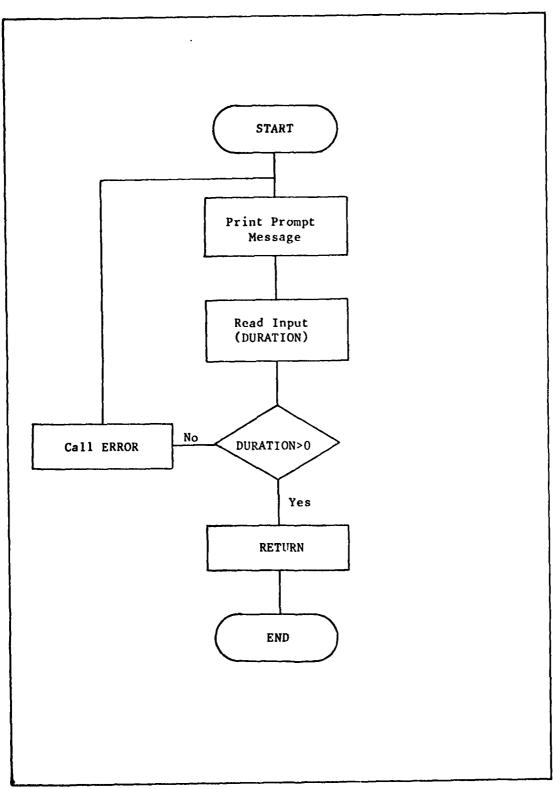


Fig 20. Flow Chart for GETTIME

```
FORTRAN IV
                      Fr1 23-Nov-79 14:56:19
                                                             PAGE 001
            V02.1-11
0001
            SUBROUTINE GETTIME ( DURATION )
      C
            TIME GETS THE DESIRED DATA PROCESSING DURATION FROM THE USER
     C AND STORES IT IN THE GLOBAL VARIABLE RTIME (USING COMMON).
     C RTIME IS AN INTEGER (LOCAL VARIABLE MINUTES), AND DURATION IS THE
      C REAL VARIABLE EQUIVALENT TO MINUTES.
        VARIABLES DEFINED HERE
     C
0002
            INTEGER*2 MINUTES
0003
            REAL*4 DURATION
            COMMON /PTIME/MINUTES
0004
       REQUEST USER TO ENTER A NEW TIME
0005
       100 TYPE 200
0006
       200 FORMAT(/2X, 'ENTER DURATION OF TEST (HOURS) EXAMPLE 15 25 <CR>'$)
0007
           ACCEPT 300, DURATION
8000
       300 FORMAT (F5.0)
0009
           IF (DURATION.LE 0.)
           1CALL ERROR ('DURATION MUST BE IN HOURS AND > .01666')
0011
            IF (ERR) GO TO 100
0013
           MINUTES=DURATION*50.
0014
           RETURN
0015
           END
```

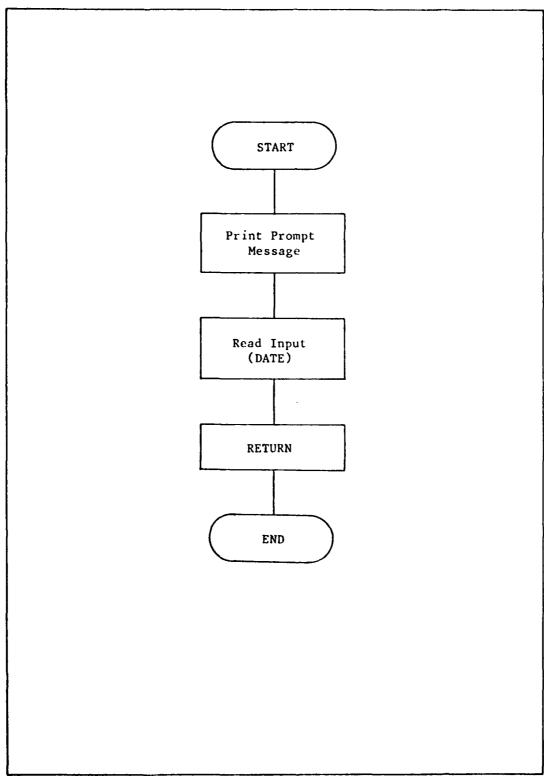


Fig 21. Flow Chart for GETDATE

```
FORTRAN IV VO2.1-11
                        Tue 27-110v-79 00:00:00
                                                             PAGE 001
0001
            SUBROUTINE GETDATE ( DATE )
      C DATE READS IN THE USER-SUPPLIED DATE AND STORES IT IN THE STRING
      C DATE. UP TO 13 CHARACTERS ARE ALLOWED. DATE IS ONLY USED IN
      C SUBROUTINE GRAPH TO PRINT OUT THE DATE.
      C
0002
            LOGICAL*1 DATE(13)
0003
            INTEGER*2 I
0004
            TYPE 100
0005
       100 FORMAT (/2X 'ENTER THE NEW DATE EX. 13-AUG-79<CR>'$)
0006
            ACCEPT 200, (DATE (I), I=1.13)
0007
       200 FORMAT (13A1)
8000
            RETURN
0009
            END
```

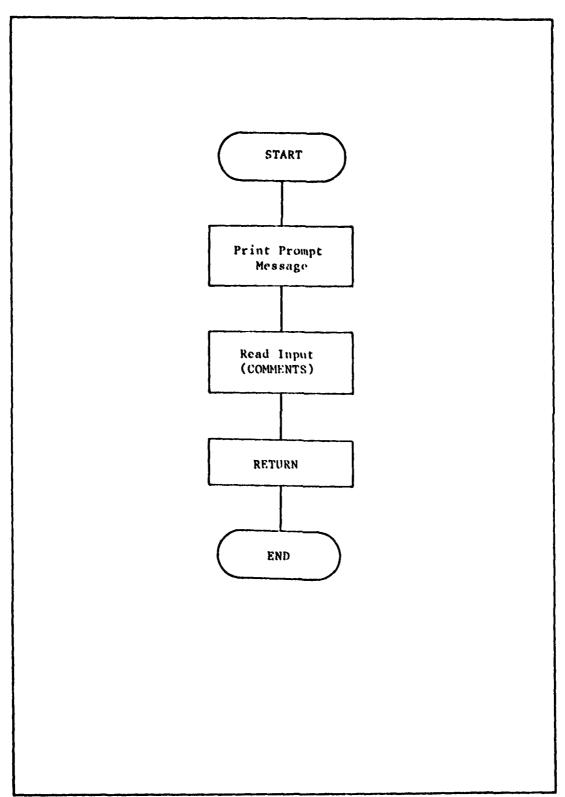


Fig 22. Flow Chart for GETCOMMENT

```
FORTRAN IV
            V02.1-11
                        Tue 27-Nov-79 00:00 00
                                                              PAGE 001
0001
            SUBROUTINE GETCOMMENT ( COMMENTS )
      C COMM READS IN USERS COMMENTS, UP TO 40 CHARACTERS
      C AND STORES THEM IN THE ARRAY COMMENTS. NO ERROR CHECKING
      C IS DONE. WHAT EVER IS ENTERED WILL BE PRINTED ON THE
      C OUTPUT TABLE.
      C
0002
            LOGICAL*1 COMMENTS (40)
0003
            INTEGER*2 I
      C
      C
0004
            DO 100, I=1,40
0005
            COMMENTS(I)=' '
0006
       100 CONTINUE
0007
            TYPE 200
0008
       200 FORMAT(/2X, 'ENTER COMMENTS - 40 CHARACTERS MAX <CR>'$)
0009
            ACCEPT 300, (COMMENTS (1). I=1.40)
       300 FORMAT(:40A1)
0010
0011
            RETURN
0012
            END
```

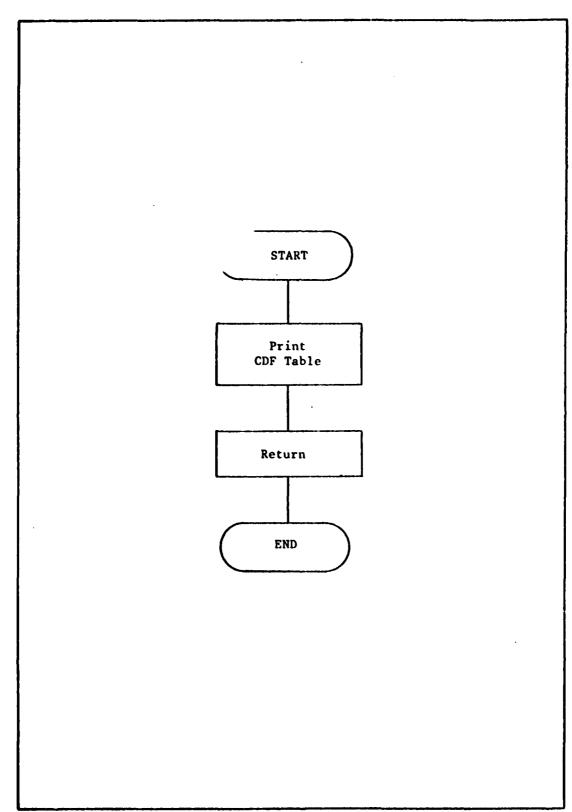


Fig 23. Flow Chart for CDFOUTPUT

```
FORTRAN IV
             V02.1-11
                         Fri 23-Nov-79 13:28:27
                                                                PAGE 001
0001
            SUBROUTINE CDFOUTPUT (CDF. NFREQ DELTAF, DATE, COMMENTS)
      C
      C
            OUTPUT IS USED TO PRINT OUT THE CDF ARRAY. THE STRUCTURE OF
      C
      C THE CDF ARRAY IS:
      C
      C
            CDF(1,1)-
                                       ;NFREQ NUMBER OF FREQUENCIES DELTAF APART
      C
      C
      C
      C
      C
            CDF (NFREQ, 1)
                              : (0-MAXFREQ)
      C
            CDF(1,2)
                              START OF TEMP INTERVAL #2
      C
            CDF(2.2)-
      C
      C
      C
      C
      C
            CDF (NFREQ, 2)
      C
      C
      C
      C
      C
            CDF(1,12)
      C
            CDF(2, 12)
      C
      C
      C
      C
      C
            CDF(120 12)
      C
      C
      C
          THE VARIABLES USED IN THIS SUBROUTINE ARE:
      C
      C
            CDF(120 12)
                                       THE CUMULATIVE DAMAGE FACTOR ARRAY
      C
            CONTENTS (40)
                              ; A STRING FOR USERS COMMENTS
      C
                              THE USER WANTS ON THE OUTPUT
      C
            DATE(13)
                              ; A STRING TO HOLD THE DATE WHICH WILL
      C
                              ;BE PRINTED ON THE OUTPUT-
      C
            DELTAF
                              ;THE FREQUENCY NFREQ BETWEEN ELEMENTS
      C
                              OF THE CDF ARRAY
      C
            FREQPAGE
                              ;THE FREQUENCY RANGE PER PAGE OF OUTPUT
      C
            IJK,L
                              GENERAL PURPOSE INDEXES
      C
            LARGE
                              ; A THREE CHARACTER ALPHA CONSTANT WHICH
      C
                               WHEN SENT TO THE PRINTER, CAUSES 80
      C
                              ; CHARACTERS PER LINE TO BE PRINTED
      C
            LOW, HIGH
                              ;TWO INDEXING LIMITS USED TO CONTROL
      C
                              THE TEMPERATURE SCALE PRINTING
                              'LOW=1=-50 F;HIGH=7=100 F
      C
      C
                              ;LOW=6=75 F . HIGH=12=225 DEG F.
      C
            HICHFREQ
                              ; INDEX LIMITS TO CONTROL THE PRINTING
```

OF THE FREQUENCY IN THE TABLE

```
C
            LOWTEMP, HIGHTEMP : INDEX LIMITS TO CONTROL THE PRINTING
      C
                              OF THE TEMPERATURE IN THE TABLE
      C
            LTSYM
                              THE SYMBOL TO THE LEFT OF THE FIRST
      C
                              :TEMPERATURE IN THE TABLE HEADING. IT IS
      C
                              ; EITHER A SPACE OR < (LESS THAN).
      C
            MAXFREQ
                              THE MAXIMUM FREQUENCY IN THE CDF ARRAY
      C
            NFREQ
                              ; THE NUMBER OF FREQUENCY COMPONENTS IN
      C
                              ;THE CDF ARRAY
      C
            NPAGE
                              ;THE NUMBER OF PAGES REQUIRED TO PRINT
      C
                              THE TABLE FOR ONE TEMPERATURE RANGE
      C
           /RTIME/RMINUTES
                              THE ACTUAL DURATION OF THE TEST RUN
      C
                              THE SYMBOL PRINTED TO THE LEFT OF THE
            RTSYM
      C
                              ;LAST TEMPERTURE ON THE TABLE HEADING.
      C
                              :IT IS EITHER A SPACE OR A > (GREATER
      C
                              ;THAN)
      C
      C
      C
            TYPE DEFINITION
      C
0002
            LOGICAL*1 DATE(13), COMMENTS(40) LARGE(4), SMALL(4), LTSYM, RTSYM
0003
            INTEGER*2 NFREQ PAGE, LOUTEMP, HIGHTEMP, I, J, K L, M N
0004
            INTEGER*2 LOW HIGH NPAGE MINUTES
0005
            REAL*4 DURATION, DELTAF, FREQ, CDF (120 12)
0006
            REAL*4 MAXFREQ HIGHFREQ FREQPAGE
      C
         GLOBAL VARIABLE
      C
0007
            COMMON /RTIME/HINUTES
      C
      C
         SMALL INITIALIZED FOR H14 PRINTER - 12 CHAR/INCH PRINTING.
      C
         LARGE - 80 CHAR/INCH.
      C
8000
            DATA SMALL/27.117.24,0/
0009
            DATA LARGE/27 117 4,0/
      C
         PROGRAM EXECUTATION STARTS HERE.
      C
0010
            DURATION=MINUTES/60.
0011
            NPAGE = (NFREQ/41) + 1
0012
            LOW = 1
0013
            HIGH=7
0014
            FREQPAGE=41.*DELTAF
0015
            LOWTEMP=-50
0016
            HIGHTEMP=100
0017
            PAGE-0
            LTSYM='<'
0018
            RTSYN=' '
0019
0020
            MAXFREQ=NFREQ*DELTAF
```

```
THE CDF TABLE FOR HALF OF THE TEMPERATURE RANGE IS NOW PRINTED OUT.
0021
            DO 150, I=1 2
0022
            FREQ=DELTAF
0023
            HIGHFREQ=0.
0024
            K=0
0025
            TYPE 80 LARGE
        PRINTING OF EACH PAGE STARTS HERE.
      C
0026
            DO 140, J=1 . NPAGE
0027
            HIGHFREQ=HIGHTREQ+FREQPAGE
0028
            IF (HIGHFREQ.GT.MAXFREQ) HIGHFREQ=MAXFREQ
0030
            TYPE 10
            FORMAT (181////)
0031
0032
            PAGE=PAGE+1
0033
            TYPE 20, PAGE
            FORMAT(24X 'CUMULATIVE DAMAGE MATRIX'13X, 'PAGE ',12/)
0034
0035
            TYPE 30, DATE
0036
       30
            FORMAT(IX 'DATE' 6X 13A1)
0037
            TYPE 35, DURATION
            FORMAT(1X DURATION', 2X, F5.2 'HOURS')
0038
0039
            TYPE 40, COMMENTS
0040
       40
            FORMAT(IX, 'COMMENTS' 2X.40A1/)
0041
            TYPE 50
0042
       50
            FORMAT(1X, FREQ (HZ)' 18X 'TEMPERATURE (DEG F)')
            TYPE 70, LTSY1, (L, L=LOWTEMP | HIGHTEMP-25 25), RTSY1, HIGHTEMP
0043
       70
0044
            FORMAT (9X.A1,5(I3.7X), I3.6X A1, I3)
0045
            TYPE 80 SMALL
            FORMAT ('+', 3A1)
0046
       80
0047
            TYPE 90
0048
       90
            FORMAT (7X, 86 ('-'))
      C
        CDF DATA IS PRINTED BY THE FOLLOWING CODE
0049
       100 IF (FREQ-GT HIGHFREQ) GO TO 130
0051
            K≈K+1
0052
            TYPE 110, FREQ, (CDF(K,N), N=LOW HIGH)
0053
       110 FORMAT(1X,F5.0.1X, '| '7(1X E10.3 1X), '| ')
0054
            FREQ=FREQ+DELTAF
            GO TO 100
0055
       130 CONTINUE
0056
         END OF PRINTING TABLE; NOW PRINT TEMP AXIS.
0057
            TYPE 90
0058
            TYPE 80 LARGE
0059
            TYPE 70.LTSYM, (L.L=LONTEMP HIGHTEMP-25, 25), RTSYM HIGHTEMP
0060
            TYPE 50
0061
            K=K-1
0062
            FREQ-FREQ-DELTAF
0063
       140 CONTINUE
```

```
SET VARIABLES TO PRINT HIGH TEMPERATURE RANGE.
0064
            LOW-6
0065
            HIGH=12
            LOWTEMP=75
0066
            HIGHTEMP=225
LTSYM='
0067
0068
0069
            RTSYM='>'
0070
       150 CONTINUE
0071
            RETURN
0072
            END
```

Appendix B

Flow Charts and Source Code for Data Processing and Execution Modules

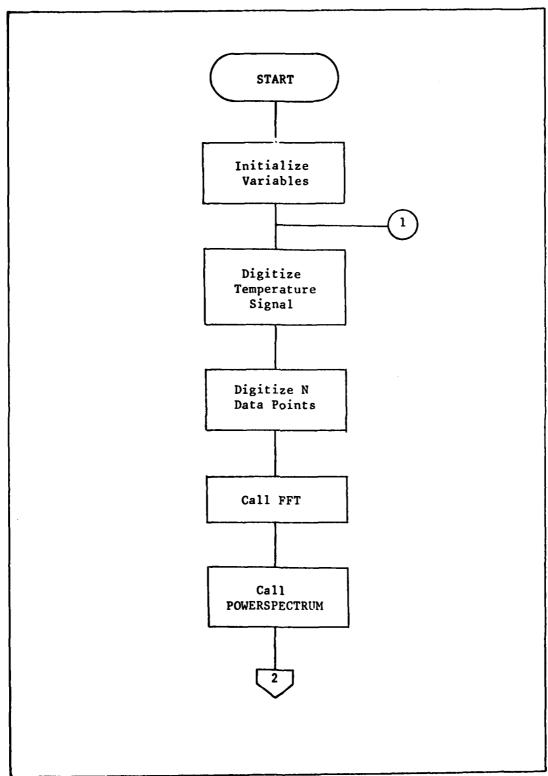


Fig 24. Flow Chart for EXECUTIVE

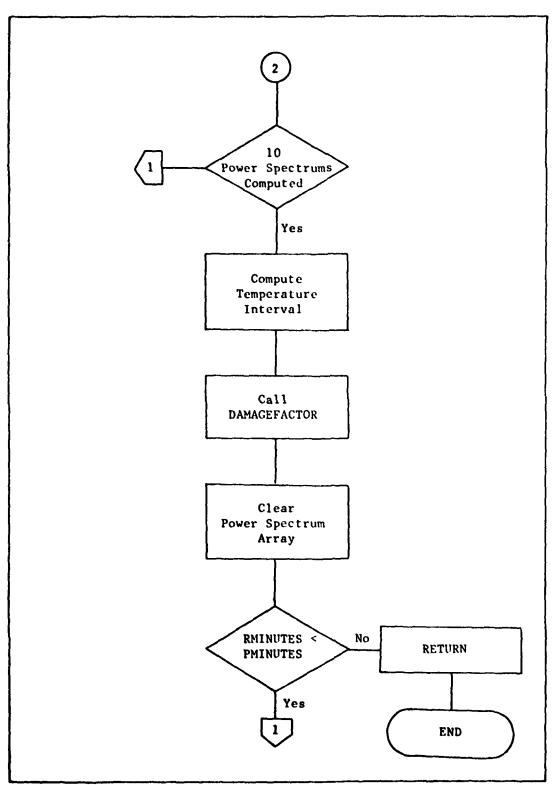


Fig 24. Flow Chart for EXECUTIVE (cont)

EXECUTIVE MACRO VO3.01 27-NOV-79 PAGE 1

	•	.TITLE EXECUTIVE
; SBTTL I	NTRODUCT	TION TO CONTROL EXECUTIVE
;		
;		
•		
;		
1. INTRO	DUCTION	TO COPTROL EXECUTIVE:
		THE CONTROL EXECUTIVE GCTS CONTROL OF
•		POUER HAS BEEN SUITCHED ON AND THE
		TCH HAS BEEN PLACED IN RUN POSITION. THE
-		RELINQUISH CONTROL IF THE POWER
-		ED OFF, THE SYSTEM HAS BEEN SUITCHED
;TO STAND	BY MODE.	OR THE RUN TIME HAS EXPIRED.
•		
·NOTE: TO	IV CLARI	DATA CTODACE ADDAC ADD DESTRED BELOW
;NOTE: TH	IE GLORL	DATA STORAGE AREAS ARE DEFINED BELOW
,		
,	CI ORI P	BITCOUNT, COUNT, INTCELNUM
		,N2.NM,NINTVL,TFLG
	-	EXEC BEGIN, RTCRATE RTCMODE TIMEINT
		ADDATA PURSPC, CDFDATA
		NDX, RTIME PTIME, ER, ADGAIN
•	"HODE IN	ADA, ALLIM CITAL, CA, APOATA
:NOTE: TH	E GLOBAT	L SUBROUTINES ARE DEFINED BELOW.

;		
	GLOBL F	FFT POWERS PECTRUM, DAMAGEFACTOR
;		The second and a more year and a second and a
;		
NOTE TH	E REGIST	TERS ARE DEFINED BELOW
;		
;		
•	MCALL .	REGDEF
	•	REGDEF

DEFINITION OF DEVICES

```
.SBTTL DEFINITION OF DEVICES
 2
        :2. DEFINITION OF DEVICES:
 3
 4
            THE DEVICES THAT ARE CALLED IN BY THE EXECUTIVE
        ; ARE DEFINED HERE.
 5
 6
 7
        ; ANALOG TO DIGITAL CONVERTOR (ADC)
 8
 9
10
11
12
        ADCSR=170400
                                 ADC CONTROL AND STATUS
13
                                 ; REGISTER.
14
        ADBFR=170402
                                 ;ADC DATA BUFFER REGISTER
15
16
        .PSECT ADGAIN, RU, D, GBL, REL, OVR
17
        ADGAIN: 0
                                 ;ADGAIN IS FOR FUTURE
18
        . PSECT
                                 :IMPLEMENTATION. IT WILL CONTAIN
                                 THE GAIN OF THE VIBRATION SIGNAL
19
20
                                 'AMPLIFIER.
21
22
        ; REAL TIME CLOCK (RTC)
23
24
25
26
        RTCSR=170420
27
                                 :RTC CONTROL AND STATUS REG
28
29
        RTBFR=170422
                                 ;RTC DATA BUFFER REGISTER
30
        ;NOTE:
31
32
33
                BEFORE THESE POINTS ARE PROCESSED, THE PROCESS
34
               LOOP COUNTER (PRCLOP) IS DEFINED
35
               AND INITIALIZED BELOW:
36
        PRCLOP: .WORD 11
37
                                 ;THIS COUNTER IS USED TO
38
                                 ;DETERMINE HOW MANY POWER
39
                                 :SPECTRUM SETS HAVE TO BE
40
                                  ADDED TOGETHER.
41
                                 ;12(OCTAL) INTERVALS 0-11.
```

INTERRUPT SERVICE ROUTINE.

```
.SBTTL INTERRUPT SERVICE ROUTINE.
        ; 3. INTERRUPT SERVICE ROUTINE:
 2
 3
         TIME INTERRUPT ROUTINE
 5
        ;AN EXTERNAL INTERRUPT (A TIME SIGNAL EVERY HINUTE)
        CAUSES A BRANCH TO THIS ROUTINE WHICH INCREMENTS
 7
 8
        :THE MINUTES COUNTER - RTIME.
 9
10
        TIMEINT: INC
                         TIME
                                          ; INCREMENT TIME BY ONE COUNT
11
                CMP
                         #74.TIME
                                          ; WHEN TIME EQUALS 60 (BASE 10)
12
                                          :INCREMENT RTIME 1.IF THE SYSTEM
13
                                           CLOCK IS SET TO 60HZ RTIME
14
                                           WILL BE IN HOURS. IF IT IS
                                           SET TO THE RTIME WILL BE IN MIN.
15
16
                BNE
                         TIM
17
                INC
                         RTIME
18
                CLR
                         TIME
                                          RESET TIME TO 0.
19
        TIM
                RTI
20
21
22
23
24
        ; 4. REAL TIME ERROR ROUTINE-
25
26
27
        ;THIS ERROR ROUTINE IS DESIGNED TO HANDLE REAL-TIME ERRORS.
28
        ; IF THE ANALOG DATA IS BEING SAMPLED TOO FAST, THIS ROUTINE
29
        ; IS CALLED. A LOCAL VARIABLE "ER" IS A COUNTER WHICH
         INDICATES THE NUMBER OF TIMES THIS ROUTINE HAS BEEN CALLED.
30
        ;IT IS FOR MAINTENCE PURPOSES ONLY. "ER" CAN BE EXAMINED
31
32
        :USING ODT.
33
34
        ERROR: MOV
                         #O GERTCSR
                                          ;STOP RTC
35
                MOV
                         @# BFR (R3)
                                         ;CLEAR A/D DONE FLAG.
36
                BIC
                         #100200. OFADCSR : CLEAR A/D ERROR FLAG.
                                          INCREMENT ERROR COUNTER.
37
                INC
                         ER
38
                BR
                         ERRTN
                                          ; RETURN
39
        TMPERR: MOV
                         (R1), R3 REMOVE BAD DATA & CLEAR DONE FLAG.
40
                                          :DONE FLAG.
                         #100000, (RO)
41
                BIC
                                          ;CLEAR ERROR FLAG.
42
                INC
                         (RO)
                                          ; RESTART CONVERSION.
43
                BR
                         LOOP
```

FPASS --FIRST PASS THROUGH THE EXECUTIVE

```
.SBTTL FPASS --FIRST PASS THROUGH THE EXECUTIVE
                .SBTTL PROCESSING CONTROL
 2
 3
        :5. PROCESSING CONTROL:
            THE TEMPERATURE SIGNAL IS SAMPLED FIRST. THE DIGITIZED
        : VALUE IS SUMMED IN A STORAGE VARIABLE "TSUM".
            THEN A SET OF N FREQUENCY DATA POINTS ARE DIGITIZED.
        THE FOURIE TRANSFORM, POWER SPECTRUM, ARE THEN COMPUTED. THIS
 9
10
        *PROCESS - ACQUIRE TEMPERATURE DATA FREQUENCY DATA, COMPUTE
11
        ;THE FFT AND THE POWER SPECTRUM - IS PERFORMED TEN TIMES FORMING
12
        :THE AVERAGE POWER SPECTRUM AND THE AVERAGE TEMPERATURE.
13
            THESE AVERAGE VALUES ARE THEN USED TO COMPUTE THE CUMULATIVE
        DAMAGE FACTOR
14
15
16
        EXEC:
                JSR
                        PC. INITAL
                                        :INITIALIZE VARIABLES
17
        BEGIN: MOV
                        O#ADBFR RO
                                        ;CLEAR AD BUFFER AND DONE FLG.
18
                MOV
                        #ADCSR,RO
                                        MOVE ADDRESS OF ADOSR AND ADBFR
19
                MOV
                        #2400.(RO)
                                        ;SET TEMP CONVERSION CH=05
20
                                        AND START TEMP CONVERSION.
21
                VOM
                        #ADBFR R1
                                        ;INTO REGISTERS FOR FASTER
22
                                        ; INSTRUCTION EXECUTION IN THE
23
                                        ;TIME CRITICAL LOOP (LOOP2)
24
                                        ; WHICH FOLLOWS.
25
                INC
                        (RO)
26
                                        ;START TEMP ACQUISITION.
       LOOP
                TSTB
                        (RO)
27
                                        ;WAIT FOR TEMPERATURE CONVERSION
28
                BPL
                        LOOP
                                        TO BE COMPLETED
29
                BIT
                        #100000,(R0)
                                        :CK FOR A/D ERROR
30
                BNE
                        TMPERR
31
               MOV
                        (R1),R3
                                        ;GET TEMP.
32
                SUB
                        #4000,R3
                                        REMOVE BIAS
33
                ADD
                        R3.TSUM
                                        ;ADD TEMP TO FORM A AVERAGE TIMES
34
                                         12(OCTAL)
35
               MOV
                        #7040, (RO)
                                        ;LOAD ADCSR- CH=16.INTERRUPT
36
                                        ;DISABLED
37
         NOTE:
        38
39
        THE ADGAIN OF THE VIBRATION SIGNAL AMPLIFIER WILL BE DETERMINED
40
41
        ;HERE. THIS HAS NOT YET BEEN IMPLEMENTED SO THE VARIABLE
42
        ;"ADGAIN" IS SET EQUAL TO 1. SO THAT IT HAS NO EFFECT ON THE
43
        :POVER SPECTRUM CALCULATION MADE IN POVERSPECTRUM
44
45
               MOV
                        #1,ADCAIN
46
47
48
49
               MOV
                       ADDATA R3
                                        ;LOAR R3 WITH LOC FOR DATA
50
               MOV
                                        ;LOAD R2 WITH THE # OF DATA POINTS
51
               MOV
                       RTCRATE, @#RTBFR : SET RT COUNTER TO -RTRATE
```

PROCESSING CONTROL

```
1
                MTPS
                         #340
                                          ;DISABLE ALL INTERRUPTS
 2
                                          ; (TIME INTERRUPTS)
 3
                MOV
                         RTCMODE, @#RTCSR ; START RTC MODE 1, 10K HZ.
        ERRTN .
        ;NOTE:
 5
 7
         THE LOOP STARTING AT "LOOP: " AND GOING THROUGH
                                                             SOB R2 LOOP 1S
 9
        ; A TIME CRITICAL LOOP. IT IS THE A/D CONVERSION ROUTINE WHICH IS
10
        FASTER THAN AN INTERRUPT ROUTINE.
        ; THE EXECUTION TIME OF THE LOOP MUST BE LUSS THAN THE AD CONVERSION
11
12
        ;TIME OR THE SAMPLING PERIOD, WHICHEVER IS SMALLER.
13
14
15
        LOOP2: TSTB
                         (RO)
                                 ; WAIT FOR AD CONVERSION
                BPL
                         3.00P2
16
                                          ;TO FINISH.
                                          ; DONE: CK FOR A CONVERSION ERROR
17
                BIT
                         #100000, (RO)
18
                BNE
                         ERROR
                                          ; ERROR GO TO ERROR ROUTINE
19
                MOV
                         (R1), (R3)
                                          ;MOVE DATA INTO DATA ARRAY
20
                SUB
                         #4000,(R3)+
                                          REMOVE 4000 BIAS AND INC R3
21
                SOB
                         R2,LOOP2
                                                  ;GO GET MORE DATE UNTILL ALL
22
                                          ;DATA POINTS HAVE BEEN PROCESSED
23
24
25
        END OF TIME CRITICAL SECTION
26
27
                CLR
                         @#RTCSR
                                          STOP RTC. THE RTC MUST BE STOPPED
28
                                           SO THE PROPER START-UP SEQUENCE
29
                                          ;STARTING AT LOOP2 IS EXECUTED.
30
                MTPS
                         #0
                                           ENABLE INTERRUPTS
31
                JSR
                         PC, FFT
        ; NOTE:
32
33
34
              AT THIS POINT, THE VALUE OF SFACTR WILL SHOW
35
              THE NUMBER OF BITS THE FOURIER TRANSFORMED ARRAY
36
              HAS BEEN SHIFTED 1-BIT TO THE RIGHT.
37
38
39
                JSR
                         PC, POWERS PECTRUM
40
```

PROCESSING CONTROL

```
1
        ;
 2
        ;
 3
 4
                TST
                         PRCLOP
                                          ;IS PRCLOP=0
 5
                BEQ
                         TEMP
                                          ; IF NOT GO BACK
                DEC
                         PRCLOP
                                          COMPUTE ANOTHER SET OF
 7
                 JMP
                         BEGIN
                                          ; POWER SPECTRUM.
 8
 9
10
        ; NOTE:
11
12
        ;BEFORE DF IS CALCULATED, THE
        ;THE TEMPERATURE INTERVAL IS
13
14
        ; DETERMINED, AND "TFLG" IS
        ;LOADED WITH THE APPROPRIATE
15
        ; VALUE -
16
17
        ;
18
19
         TEMP:
                MOV
                         #TINTVL, R1
20
                MOV
                         #13,R2
21
                MOV
                         #1 TFLG
22
                CMP
                         TSUA, (R1)+
         IN3:
23
                BLE
                         IN5
24
                         TFLG
                INC
25
                SOB
                         R2, IN3
26
         IN5:
                BR
                         DAMGE
27
28
        ; INTVL
                    1
                           2
                                  3
29
                164413,170452,174561 000737
        TINTVL
30
        ;TEMP:
                 -37.5 -12.5 12.5 37.5
31
        ; INTVL
                    5
                                 7
                           6
                 005166,011466,016034,022453
32
                 62.5 87.5 112.5 137 5
33
        ;TEMP:
34
        ;INTVL: 9
                        10
                               11
                                      12
35
                027022,033371 040010.044427
36
        ;TEMP: 162.5 187.5
                                212.5
                                         237.5
```

PROCESSING CONTROL

```
; PROCESSING CONTROL CONTINUED.....
 2
 3
        DAMGE: JSR
                         PC DAMAGEFACTOR COMPUTE DF
 6
        ; NOTE: ONCE THE DF IS COMPUTED, THE WHOLE LOOP
         :- --- IS REPEATED AGAIN. THE FOLLOWING CLEARS
 9
                THE REQUISITE VARIABLES AND ARRAYS.
10
11
        ;
12
                CLR
                         TSUM
13
                MOV
                         #11 PRCLOP
14
15
                MOV
                         #PWRSPC RO
                                          THE POWER SPECTRUM
16
                                           IS NOW CLEARED.
17
                MOV
                         NINTVL,R1
                                          ;LOAD RO WITH THE
18
                                          STARTING ADDRESS AND
19
                                          R1 WITH ITS SIZE-
20
21
        CLEAR: CLR
                         (R0)+
22
                SOB
                         R1, CLEAR
23
24
        ;PTIME IS A GLOBAL VARIABLE INITIALIZED IN
        ;SUBROUTINE SETUP IN MFDM. IT IS THE TIME IN MINUTES.
25
26
        ; FOR WHICH THE TEST SHOULD RUN. RTIME IS A GLOBAL
27
        ; VARIABLE WHICH IS THE ACTUAL RUN TIME RETURNED
28
         TO MFDM .
29
        ;RTIME IS NOW CHECKED AGAINST PTIME AND
30
        ; IF RTIME IS LESS THAN PTIME PROCESSING CONTINUES
31
        ;ELSE CONTROL IS RETURNED TO MFDM OR THE SYSTEM
32
        ; HALTS (IF RUNNING WITH OUT A TERMINAL).
33
34
                CMP
                         PTIME, RTIME
                                         ; IF PLANNED RUN TIME
                BGT
35
                         BEGIN
                                          ;> RUN TIME CONTINUE
36
                RTS
                         PC
                                          ;ELSE RETURN
37
38
39
        END OF PROCESSING CONTROL
```

DATA STRUCTURE AND VARIABLES USED

```
.SBTTL DATA STRUCTURE AND VARIABLES USED
 1
 2
 3
 4
        · 6. DATA STRUCTURE AND VARIABLES USED:
 5
        ; THE DATA STRUCTURE WHICH IS USED TO STORE ACQUIRED
 6
 7
        ;DATA AND DIFFERENT VARIABLES USED ARE GIVEN BELOW:
 8
 9
10
        .PSECT NINTVL RW, D, GBL, REL OVR
11
                NINTVL . WORD
12
        .PSECT TFLG,RW,D,GBL,REL,OVR
13
                TFLG:
                       WORD
14
        .PSECT ADDATA RW D,GBL,REL,OVR
15
                DATALOC . REPT
                                400
16
                                 0
                        .WORD
17
                        • ENDM
18
19
                ADDATA: DATALOC
20
21
        .PSECT PTIME RW,D,GBL,REL,OVR
22
                PTIME
                       •WORD O
23
```

INITIALIZE

```
1
         .SBTTL INITIALIZE
         . PSECT
 2
 3
         :1. DESCRIPTION
 8
         THIS SUBROUTINE IS USED TO INITIALIZE VARIOUS
 9
         : PARAMETERS USED IN EXEC AND FOURIE. TWO PARAMETERS
         : (B AND INDX) ARE INITIALIZED BY SUBROUTINE SETUP WHICH
10
11
         ; IS CALLED BY MFDM. THESE TWO VARIABLES ARE USED TO
12
         ; INITIALIZE THE REST OF THE VARIABLES IN THIS ROUTINE.
13
14
         THE VARIABLES WHICH ARE INITIALIZED IN THIS
15
         ;SUBROUTINE ARE:
16
17
         ;BITCOUNT
                         A COUNTER USED IN THE FOURIE
18
                         BIT INVERSION ROUTINE IT IS
19
                         EQUAL TO THE NUMBER OF BITS
20
                         IN THE NUMBER - (2**B)-1.
21
                         OR BITCOUNT=B-1.
22
        ;COUNT
23
                         A COUNTER USED IN FOURIE AND
24
                         INITIALIZED TO HALF THE NUMBER
25
                         OF COMPLEX DATA POINTS.
26
                         POSSIBLE VALUES ARE 64 32,16,8
27
28
         ·ER
                         A COUNTER USED IN EXEC WHICH
29
                         CONTAINS THE NUMBER OF TIMES
30
                         THE ERROR ROUTINE WAS ENTERED
31
32
        ; INTCELNUM
                         AN INITIAL VALUE FOR "CELNUM"
33
                         WHICH IS USED IN FOURIE-
34
                         "2ND PASS AND ONWARD" ROUTINE.
35
                         POSSIBLE VALUES ARE 32,16,8,4
36
37
        ;N
                         THE NUMBER OF DATA POINTS TO BE
                         PROCESSED. USED IN EXEC AND
38
39
                         FOURIE POSSIBLE VALUES ARE
40
                         256, 128, 64, 32.
41
42
        :N2
                         THE NUMBER OF "COMPLEX" DATA POINTS
                         USED IN FOURIE POSSIBLE VALUES
43
44
                         ARE 128 64,32,16.
45
                        AN INDEX USED IN FOURIE- POST
46
        ; NM
47
                         PROCESSING" ROUTINE. IT IS
48
                         INITIALIZED TO THE RELATIVE
49
                        LOCATION OF THE N-1 COMPLEX
50
                        DATA POINT. POSSIBLE VALUES ARE
51
                         774 374.174 74.
```

INTRODUCTION

```
1
 2
         ; 2. INITIALIZATION
 3
 4
 5
         INITAL
                 MOV
                          #TIMEINT 0#100
                                            PUT INTERRUPT ADDR. IN LOC 100
 6
                 MOV
                          #0,0#102
                                            ;INTERRUPT PRIORITY=0
 7
                 MOV
                          #1.R0
                                            :N=2**B
 8
                 ASH
                          B RO
 9
                 MOV
                          RO, N
10
11
                 MOV
                          RO, NM
                                            ;NM=2(N-2)
12
                 SUB
                          #2. NM
13
                 ASL
                          NM
14
15
                 ASR
                          R0
                                            ·N2=N/2
16
                 MOV
                          RO, N2
17
18
                 ASR
                          R0
                                             COUNT=N2/2
19
                 MOV
                          RO. COUNT
20
21
                 ASR
                          R0
                                            :INTCELNUM=COUNT/2
22
                 MOV
                          RO, INTCELNUM
23
24
                 CLR
                          ER
                                            :ER=0
25 ·
                 CLR
                          RTIME
                                            ; INITIALIZE RUN TIME
26
                 CLR
                          TIME
                                            ;AND THE TIME COUNTERS.
27
28
                 HOA
                          B. BITCOUNT
                                            ;SET BITCOUNT=B-1
29
                 SUB
                          #1,BITCOUNT
30
                 CLR
                          TSUM
31
        ;
32
33
                 MOV
                          #PWRSPC,RO
                                            THE POWER SPECTRUM
34
                                            IS NOW CLEARED
35
                 MOV
                          #360,R1
                                            ;LOAD RO WITH THE STARTING
36
                                            AND RI WITH THE SIZE OF
37
        CLEAR1: CLR
                          (R0)+
                                            ;THE ARRAY (127 DECIMAL).
38
                 SOB
                          R1, CLEAR1
                                            ;CLEAR THE ARRAY.
39
                 MOV
                          #CDFDATA, RO
                                            ; REPEAT FOR THE CDF ARRAY.
40
                 MOV
                          #5500,R1
41
        CLEAR2: CLR
                          (R0)+
42
                          R1, CLEAR2
                 SOB
43
                 RTS
                          PC
```

DEFINITION OF DEV

```
1
                  .SHTTL DEFINITION OF DEV
 2
         . PSECT RTCRATE, RW, D, GBL. REL OVR
 3
         RTCRATE: . WORD O
                                    ; REAL TIME CLOCK OVERFLOW
                                    ;RATE CONTROL. OVERFLOW SIGNAL
                                     RATE
                                            CLOCK FREQUENCY/RTC-RATE
 3
         . PSECT RTCMODE RW.D.GBL.REL, OVR
 Q
10
         RTCMODE: . WORD 0
                                    REAL TIME CLOCK SETTING FOR
11
                                    :THE RTC CSR.
12
         TSUM
                  .WORD 0
13
         TIME:
                  .WORD O
14
                  .WORD 0
15
         . PSECT N2, RV, D GBL, REL, OVR
16
         N2:
                  .WORD 0
17
         NM
                  .WORD 0
18
                 .WORD 0
         COUNT:
19
         INTCELNUM: .WORD O
20
         .PSECT ER, RV, D, GBL, REL, OVR
21
                  -WORD O
22
         . PSECT B RW D. GBL, REL, OVR
23
         B:
                  .WORD O
24
         BITCOINT: .WORD O
25
         . PSECT INDX, RU, D, GBL, REL, OVR
26
         INDX:
                  .WORD 0
27
         . PSECT RTIME RW, D, GBL, REL, OVR
28
        RTIME
                  .WORD O
29
         . PSECT PWRSPC RU, D, GBL, REL, OVR
30
        PWRSPC: . REPT
                          360
31
                  .WORD
                          0
32
                  . ENDM
33
         .PSECT CDFDATA, RN.D. GBL, REL, OVR
34
35
        CDFDATA: .REPT
                         5500
36
                 .WORD
37
                 . ENDM
38
                 • END
```

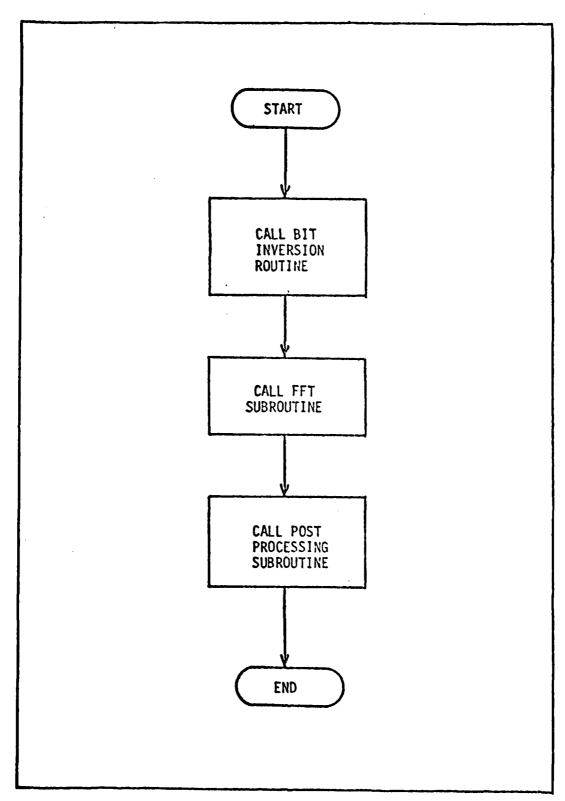
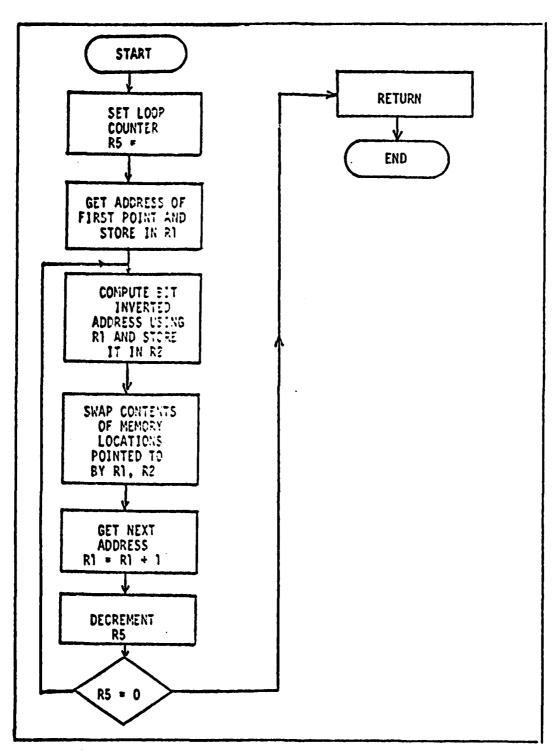


Fig 25. Flow Chart for FFT (Ref 7:78-90)



1

Fig 25. Flow Chart for FFT (cont) - Bit Inversion Routine

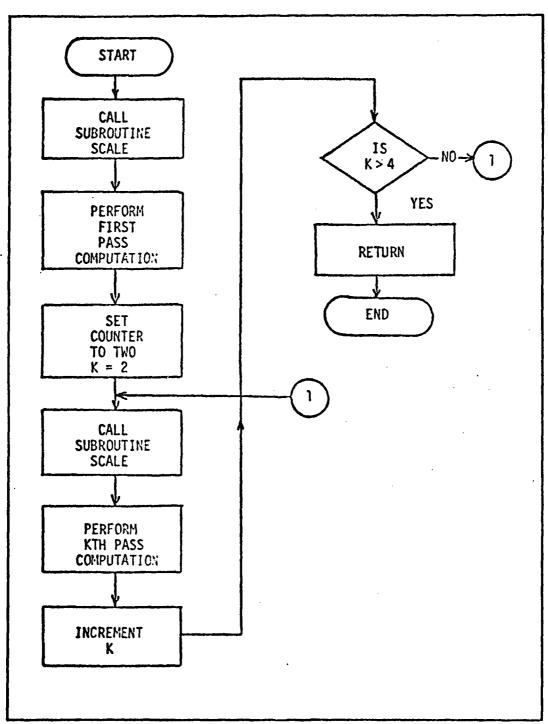


Fig 25. Flow Chart for FFT (cont) - Fast Fourier Transform

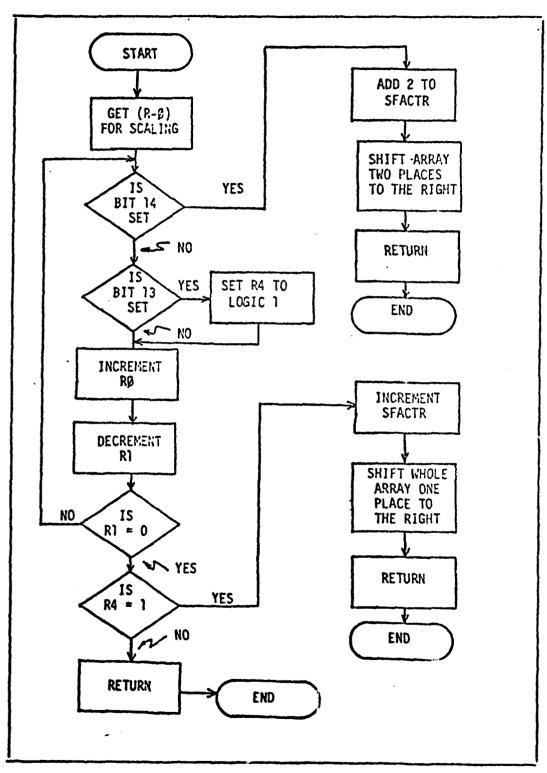


Fig 25. Flow Chart for FFT (cont) - Scale Routine

FFT --FFT SUBROUTINE MACRO VO3.01 28-NOV-79 PAGE 1

```
1
 2
        .TITLE FFT --FFT SUBROUTINE
        .SBTTL INTRODUCTION --INTRO TO FFT AND ITS USE.
 3
        .CSECT
 5
 6
        ; 1. DESCRIPTION:
 7
 9
             THIS SUBROUTINE CAN BE USED TO TRANSFORM "N" REAL
10
       ; DATA POINTS. IT ASSUMES THE DATA EQUIVALENT TO "N2"
11
       ; COMPLEX POINTS (THE ODD REAL POINTS ARE CONSIDERED
12
        ; REAL PARTS AND THE EVEN PARTS ARE CONSIDERED COMPLEX)
        ; AFTER THE TRANSFORMATION, A POST PROCESSING OF DATA GIVES
13
        ; "N2" FOURIER TRANSFORMED FREQUENCY COMPONENTS (THE OTHER
14
        ; "N2" COMPONENTS ARE THE COMPLEX CONJUGATES OF THESE
15
16
        · VALUES).
17
18
        ; 2. GLOBAL SYMBOLS:
19
20
                THE FOLLOWING SYMBOLS OF "FFT" ARE INITIALIZED
21
22
        ; OR USED IN OTHER ROUTINES THAT ARE CALLED BY EXEC-
23
24
         .GLOBL FFT ADDATA.N.N2 NM, COUNT, INTCELNUM, INDX
25
         .GLOBL BITCOUNT, SFACTR
26
27
        ; 3. REGISTER DEFINITION
28
29
30
31
         .MCALL .REGDEF
         . REGDEF
32
33
34
35
```

INTRODUCTION -- INTRO TO FFT AND ITS USE.

1	; 4. DEFINITION OF	PARAMETERS USED	
7		ARE THE DIFFERENT PARAMETERS USED TINE "FFT" WHICH ARE DEFINED	
6 7	;		
8	; N	N IS THE # OF REAL DATA POINTS IN OCTAL FORMAT	
10 11 12	; ; N2:	N2 IS THE ASSUMED # OF COMPLEX POINTS IN OCTAL.	
13	; ; ;	B IS THE EXPONENT OF 2	
15 16	;	WHICH SATISFIES THE RELA -TIONSHIP N2=2**B.	
17 18 19	; ADDATA	ADDATA IS A POINTER THAT WILL CONTAIN THE STARTING ADDRESS	
20 21	;	OF THE INPUT (REAL) POINTS THAT ARE TO BE TRANSFORMED.	
22 23 24	; COUNT:	A COUNTER SET TO HALF THE NUMBER OF COMPLEX POINTS.	
25 26	;		
27 28 29	5. DATA STORAGE A	REA ALLOCATION:	
30	-	T "ADDATA" IS DEFINED IN EXEC NG ADDRESS OF AN "N" WORD ARRAY	
32 33 34	; WHICH CONTAINS THE DIGITIZED DATA AND WILL ; CONTAIN THE TRANSFORMED DATA		
35	;		

FFT --DIFFERENT PARTS AND THEIR CODE.

```
.SBTTL FFT -- DIFFERENT PARTS AND THEIR CODE.
        ; 1. THE BIT INVERSION ALGORITHM
 2
 3
 4
            THE INPUT DATA POINTS ARE BIT INVERTED (OR SHUFFLED) SO
 5
            THAT THE FINAL TRANSFORMED ARRAY IS IN CORRECT ORDER.
 6
 7
        FFT
                 CLR
                         SFACTR
                                           ;SET SCALE FACTOR TO O
 8
                 CLR
                         RO
                                           RO WILL COUTAIN THE
 9
                                           STARTING ADDRESS OF DA
10
                                            -TA TO BE TRANSFORMED.
11
                 CLR
                         RI
                                           ;RI WILL CONTAIN BIT
12
                                           ; INVERTED ADDRESS.
13
                 MOV
                         N2 R5
                                           COUNTER TO KEEP TRACK
14
                                           OF # OF DATA POINTS.
15
        BIV10:
                MOV
                         BITCOUNT.R4
                                           ;BIT SHIFTING COUNTER
16
                 CLR
                         R 2
17
        BIV20:
                 CLC
                                           CARRY PLAG IS CLEARED
18
                                           BECAUSE BITS WILL BE
19
                                           ;COPIED INTO C-FF BY
20
                                           ROTATATIONS, WHICH WILL
21
                                           ;ACCOMPLISH SHIFT FROM R1
22
                                           ; INTO R2 OF LS3.
23
                 ROR
                         R1
24
                 ROL
                         R2
25
                 SOB
                         R4.BIV20
                                           AT THIS POINT 4 LSB OF
26
                                           (RI ARE COPTED INTO R2
27
                                           IN BIT REVERSED ORDER.
28
                 MOV
                         RO,R1
                                           COPY FOR SWAP
29
                 CMP
                         R2, R1
                                           ; COMPARE TO PREVENT
30
                                           ; DOUBLE SWAP.
31
                 BLT
                         BIV30
32
                 ASL
                                           ;LEFT SHIFT RI FOR
                         R1
33
                                           ;BYTE ADDRESSING.
34
                 ASL
                         R1
35
                         R2
                 ASL
36
                 ASL
                         R2
37
                 ADD
                                          STARTING ADDRESS
                         ADDATA, R2
38
                 ADD
                         ADDATA R1
                                          :DESTINATION ADDRESS
39
                 JSR
                         PC SWAP
40
                 CMP
                         (R1)+, (R2)+
                                          :ADVANCE TO IMAG PART
41
                         PC SWAP
                 JSR
                                           SWAP DATA
        BIV30:
42
                 INC
                         RO
                                          ;GO TO NEXT POINT.
43
                 MOV
                         RO,R1
                                          COPY RO.R1 FOR
44
                                           INVERION.
45
                 SOB
                         R5 BIV10
        ; NOTE:
46
47
48
                 AT THIS POINT THE BIT INVERSION IS COMPLETE
49
                 AND DATA IS READY TO BE TRANSFORMED.
```

FFT -- DIFFERENT PARTS AND THEIR CODE.

```
; 2. FAST FOURIER TRANSFORMATION (FIRST PASS):
 2
 3
           DURING THIS PASS THERE ARE NO MULTIPLICATIONS, AND
          THE FOLLOWING EQUATIONS ARE IMPLEMENTED
 5
                R(M)'=R(M)+R(N)
 7
                I(M)'=I(M)+I(N)
 8
                R(N)'=R(M)-R(N)
                 I(N)'=I(M)-I(N)
 9
10
11
                                           ;LOAD STARTING ADDRESS
        FPASS: MOV
                         ADDATA RO
12
                                           OF DATA FOR SCALING.
13
                MOV
                         NR1
                                           :LOAD # OF POINTS.
14
                         PC, SCALE
                                           ALWAYS SCALE DATA TO
15
                 JSR
16
                                           ; PREVENT OVERFLOW.
17
                MOV
                         ADDATA RO
18
                MOV
                         RO,R1
                                           ;FIRST POINT R(M)
19
                CMP
                          (R1)+,(R1)+
                                           2ND POINT R(N)
                         COUNT, R5
20
                MOV
                                           ; COUNTER=N2/2
21
        PASS1:
                MOV
                          (RO) R2
                                           ; SAVE R(M)
22
                                           ;R(M)'=R(M)+R(N)
                          (R1), (R0)+
23
                 ADD
24
                 SUB
                          (R1), R2
                                           R(N)'=R(M)-R(N)
25
                MOV
                         R2, (R1)+
                          (RO), R2
26
                MOV
27
                 ADD
                          (R1), (R0)+
                                           (N)I+(M)I=(M)I;
                          (R1), R2
28
                 SUB
                                           ;I(N)'=I(M)-I(N)
                         R2, (R1)+
29
                MOV
30
                          (R0)+,(R1)+
                 CMP
31
                CMP
                          (R0)+, (R1)+
                                           GO TO NEXT PAIR
32
                         R5, PASS1
                SOB
                                           ;GO BACK N2/2 TIMES
                         SPASS
33
                BR
34
        ; NOTE:
35
36
                FIRST PASS IS OVER NOW GO TO SECOND PASS.
37
```

FFT -- DIFFERENT PARTS AND THEIR CODE

1 2	; 3. FAST FOURI	ER TRAN	ISFORM (2ND PASS AND ONWARD):
3	THE 2ND 3R	D AND 4	ATH PASS USE THE SAME LOOP,
4			THE PARAMETERS DEFINED
5	; BELOW TAKE DI		
6	;		
7	DELTAY	0	;SIZE OF INCREMENT FOR EXPO
8	;		-NENT OF "W" WHICH SATISF
9	;		-IES THE FOLLOWING RELATIO
10	;		-NSHIP:
11	;		W=EXP(2*PI*K/N)
12	;		K=0.1,2N-1
13	;		N= # OF POINTS
14	;		
15	CELNUM:	0	;# OF CELLS IN EACH PASS.
16	;		
17	PAIRNM:	0	; # OF PAIRS IN EACH PASS.
18	;	_	
19	CELDIS:	0	DISTANCE (BYTES) BETWEEN A PAIR OF
20			; POINTS IN A CELL.
21	;	_	
22	POINT.	U	;TEMPORARY POINTER TO STARTING
23	;		ADDRESS OF DATA TO TRANSFORMED.
24	. ARI AND	^	ARLY ACTIVITIES NO STRAIN AND ARLANG
25	CELCNT.	U	CELL COUNTER TO KEEP TRACK
26 27	;		OF # OF CELLS IN EACH PASS.
27 28	;	^	. CONTROL TO THE TOTAL OF
29	PARCNT:	U	COUNTER TO KEEP TRACK OF
30	;		# OF PAIRS IN EACH CELL.
31	; SINE:	0	TEMPORARY STORAGE TO HOLD
32		J	VALUE FOR SIN(Y).
33	; ;		AUDOF LOW STULLY.
34	cos.	0	TEMPORARY STORAGE TO HOLD
35	;	V	VALUE FOR COS(Y).
36	; ;		**************************************
37	, Υ	0	THIS INDICATOR HOLDS THE
38	;	•	VALUE OF "Y" BEFORE SINE/
39	<i>;</i> •		COSINE LOOK UP TABLES ARE
40			SEARCHED.
41	;		
42	INTCELN	UM:	THE INITIAL VALUE FOR "CELNUM"
43	;		WHICH IS DEFINED IN EXEC.

FFT --DIFFERENT PARTS AND THEIR CODE

```
; 4. CODE FOR 2ND PASS AND ONWARD.
 2
 3
                THE EQUATIONS THAT ARE IMPLEMENTED IN 2ND 3RD
        ; AND 4TH PASS ARE GIVEN BELOW:
 4
 5
 6
                R1'=R1+R2*COS(Y)+I2*SIN(Y)
 7
                I1'=I1-R2*SIN(Y)+I2*COS(Y)
 8
                R2'=R1-R2*COS(Y)-I2*SIN(Y)
 9
                12'=11+R2*SIN(Y)-12*COS(Y)
10
        ; WHERE Y=2*PI*K/N2, N2=# OF COMPLEX POINTS, AND
11
12
        K=0,1,2... N2/2-1
13
14
        ; NOTE:
15
16
17
                3RD AND 4TH PASSES START FROM LABEL "NEWPASS"
18
                WHEREAS 2ND PASS STARTS AT LABEL "SPASS".
19
20
        SPASS: MOV
                         #200, DELTAY
                                          ·INITIALIZE DELTAY TO
21
                                          ;HALF THE SIZE OF THE
22
                                          ;LOOK-UP TABLE
23
24
                NOV
                         INTCELNUM, CELNUM
25
                MOV
                         #2 PAIRNY
26
                MOV
                         #10, CELDIS
27
28
        NEWPAS: MOV
                         ADDATA RO
                                          GET STARTING ADDRESS
29
                                          AND # OF POINTS BEFORE
30
                                          'CALLING SUB:"SCALE"
31
                MOV
                         N,R1
32
                JSR
                         PC, SCALE
33
34
                MOV
                         CELNUM, CELCUT
                                          'SET UP CELL COUNTER.
35
                MOV
                         ADDATA POINT
36
                BR
                         NEWCEL
```

FFT --DIFFERENT PARTS AND THEIR COMM

```
:NOTE: EACH NEW CELL ST (IN A PASS STARTS HERE.
 1
 2
                SOME TEMPORARY STORAGE VARIABLES ARE DEFIN
 3
                -ED BELOW:
 5
                 12SINY
                                  ;12*SIN(Y) IS STORED HERE.
                          0
 6
                 R2SINY
                                  ;R2*COS(Y) IS STORED HERE.
 7
                 I2COSY: 0
                                  ;12*COS(Y) IS STORED HERE.
                 R2COSY: 0
                                   ;R2*COS(Y) IS STORED HERE.
 8
 9
10
        NEWCEL
                MOV
                          PAIRNM PARCNT
                                            CLEAR Y
11
                 CLR
12
        NEWC10: MOV
                          POINT, RO
13
                 MOV
                          RO RI
                          CELDIS R1
14
                 ADD
15
                 MOV
                          Y.R2
                                           ;LOAD REG 2 WITH THE VALUE
16
                                           TO GET THE SIN & COS FOR-
17
                 MOV
                          SINTAB(R2), SINE 'GET THE VALUE OF THE
18
                                            SINE(R2)
19
                 MOV
                          COSTAB(R2) COS
                                           ;COS (R2)
20
                 MOV
                          (R1),R2
                                           ;GET REAL PART R2
21
                 MUL
                          COS,R2
                                           ;R2*COS(Y).
22
                          #1,R2
                 ASHC
                                           ; LEFT SHIFT FOR FRACTIO
23
                                           :-NAL MULTIPLICATION.
24
                 MOV
                         R2, R2COSY
                                           ;R2COSY=R2*COS(Y).
25
                 MOV
                          (R1), R2
                                           GET REAL PART R2.
26
                 MUL
                          SINE R2
                                           ;R2*SIN(Y)
27
                          #1 R2
                 ASHC
28
                 HOV
                          R2, R2SINY
                                           ;R2SINY=R2*SIN(Y).
29
                 MOV
                          R1,R4
                                           COPY PRESENT ADDRESS.
30
                 TST
                          (R4)+
                                           GET ADDRESS OF IMAG.
31
                 MOV
                                           ;THIS IS VALUE OF 12.
                          (R4),R2
32
                 MUL
                          SINE, R2
                                           ; 12*SIN(Y).
33
                 ASHC
                          #1,R2
                         R2 I2SINY
34
                 MOV
                                           ; I 2SINY=I 2*SIN(Y).
35
                MOV
                          (R4), R2
36
                         COS R2
                 MUL
                                           ; I2*COS(Y).
37
                 ASHC
                          #1,R2
38
                 MOV
                         R2 I2COSY
                                           ; I2COSY=I2*COS(Y).
```

FFT -- DIFFERENT PARTS AND THEIR CODE.

```
; NOTE:
 1
 2
        ; ----
                NOW COMBINE TERMS TO GET THE VALUES FOR
 3
         ; R1'.R2', I1' ANDI2'.
 6
                 MOV
                          (R0), (R1)
                          R2COSY (RO)
                                           ; (R0)=R1+R2*COS(Y)
                 ADD
                          I2SINY (RO)
 8
                 ADD
                                           ;R1'=R1+R2*COS(Y)+I2*SIN(Y).
                          12814Y. (R1)
 9
                 SUB
                                           ; (R1) = R1 - I2 \times SIN(Y).
10
                 SUB
                          R2COSY. (R1)
                                           ;R2'=R1-T2*SIN(Y)-R2*COS(Y).
11
                 MOV
                          R4 R1
12
                          (R0)+
                 TST
13
                          (R0), (R1)
                 MOV
                          R2SINY (RO)
14
                 SUB
                                           (R0)=11-R2*SIN(Y).
15
                 ADD
                          I2COSY (RO)
                                           ; I1'=I1-R2*SIN(Y)+I2*COS(Y).
                          R2SINY (R1)
16
                 ADD
                                           ;(R1)=I1+R2*SIN(Y).
17
                          I?COSY (R1)
                 SUB
                                           ;12'=11+R2*SINY-12*COS(Y).
18
19
        ; NOTE:
20
21
22
                ONE PAIR OF POINTS HAS BEEN DONE NOW DO
23
                NEXT PAIR
24
25
26
                          #4 POINT
                                           NEXT ADDRESS IN THE CELL.
                 ADD
27
                 ADD
                          DELTAY, Y
                                           ; INCREMENT EXPONENT OF W.
28
                 DEC
                          PARCN'T
                                           ONE PAIR DONE.
29
                 BNE
                          NEWC10
                                           ;DO NEXT PAIR EXIT IF ALL
30
                                            PAIRS WITHIN A CELL ARE
31
                                           ; DONE.
32
33
34
        ; NOTE:
35
                ONE CELL DONE NOW RESET EXPONENT OF W AND
36
37
               PAIR .- COUNTER (PARCNT).
38
39
40
                 ADD
                         CELDIS POINT
                                           ;ADDRESS OF NEXT CELL.
41
                 DEC
                         CELCAT
                                           ;ALL CELLS DONE?
42
                 BNE
                         NEWCELL
                                           :NO,GO BACK.
```

FFT -- DIFFERENT PARTS AND THEIR CODE.

t

1	; NOTE:	:		
2	;	•		
3	;	ONE PAS	s is done no	W GO ON TO NEXT PASS,
4	•	SET NEW	VALUES OF P	ARAJETERS DEFINED
5	;	IN PARA	3.	
6	•			
7	•	ASR	CELNUM	;LESS CELLS THIS TIME
8		BEQ	POST	;ALL DONE WHEN VARIABLE
9		•		;CELNUM=0.
10		ASL	PAIRNM	THIS TIME THERE ARE
11				MORE PAIRS IN EACH CELL.
12		ASL	CELDIS	PAIRS ARE FURTHER APART.
13		ASR	DELTAY	LESS INC OF Y PER CELL.
14		BR	NEWPAS	GO BACK AND DO NEXT PASS
15	;			·
16	;			
17	•			
18	r			
19	:			
20	•			
21	END C	F FFT AL	GORITHM.	

```
.SBTTL POST PROCESSING ALGORITHM
 2
 3
 4
          1. INTRODUCTION TO POST PROCESSING ALGORITHM
 6
             POST PROCESSING OF THE TRANSFORMED ARRAY IS DONE
 7
        ; BECAUSE THE INPUT (REAL) DATA WAS ASSUMED TO BE CONPLEX
 8
 9
        ; AND THE OUTPUT OF THE FFT IS NOT IN CORRECT FORM.
10
              THE EQUATIONS THAT ARE IMPLEMENTED IN THIS PASS
        ; ARE GIVEN BELOW:
11
12
13
                AR(M) = (RP+IP*COS(T)-RM*SIN(T))/2
14
                AI(M)=(IM-IP*SIN(T)-R!1*COS(T))/2
15
                AR(N-M)=(RP-IP*COS(T)+RM*SIN(T))/2
16
                AI(N-M)=(-IM-IP*SIN(T)-RM*COS(T))/2
17
18
        ; WHERE T=PI*M/N, N=16 AND M=0.1,2.
        ; RP=I(M)+I(N-M), RP=R(M)+R(N-M), IM=I(M)-I(N-M) AND
19
20
        ; RM=R(M)-R(N-M).
21
22
              SOME TEMPORARY VARIABLES THAT ARE USED IN
        ; THIS PASS ARE DEFINED BELOW-
23
24
        ;
25
                RP:
                         0
26
                IP:
                         0
27
                RM:
                         0
28
                IM:
                         0
29
                ARM
                                 ; ARM=AR (M)
30
                ARIPI
                         0
                                 ; ARNM=AR(N-M)
31
                AIM
                         0
                                 ;AIM=AI(M)
32
                AINM:
                         0
                                 ;AINM=AI(N-M)
33
                INDX:
                                 INDEX TO GET SIN, COS VALUES.
34
                                 (INITIALIZED IN SETUP)
35
                NM
                                 A RELATIVE OFFSET EQUAL TO
36
                                 THE N-1 COMPLEX DATA POINT
37
                                 NM=2(N-2) AND IS INITIALIZED
38
                                 IN EXEC.
```

```
; 2. CODE FOR POST PROCESSING :
 2
 3
          NOTE:
 5
 6
                SET UP ADDRESS POINTERS AND OTHER RELATED
 7
                COUNTERS.
 9
 9
10
        POST:
                NOV
                         ADDATA RO
                                          :SET ADDRESS POINTER
11
                                          ;AND # OF POINTS BEFORE
12
                                          ;SCALING DATA ARRAY.
13
                MOV
                         N,R1
14
                 JSR
                         PC, SCALE
15
                MOV
                         ADDATA RO
16
                MOV
                         COUNT R1
                                          ;SET COUNTER TO DECIMAL 8.
17
                                          ; INCREMENT FOR SINE POINTER.
                NOV
                         INDX R5
18
                MOV
                         NM R4
                                          OFFSET FOR ADDRESS OF
19
                                           POINT.
20
                ADD
                         ADDATA.R4
21
22
23
        ; NOTE:
24
25
               THE POST PROCESSING OF FIRST POINT IS DONE
26
               SEPARATELY BECAUSE WE DO NOT PHYSICALLY GE
27
               -NERATE THE (N+1)ST POINT.
28
29
30
        POST1: MOV
                         2(RO) R2
                                          GET IMAG PART OF 1ST
31
                                          :POINT.
32
                ADD
                         R2 (R0)
                                          ;ADD IT TO REAL PART.
33
                ASL
                         (R0)+
                                          ;DOUBLE THIS RESULT TO
34
                                           MAINTAIN SCALE.
35
                CLR
                         (R0)+
                                          CLEAR THE LOCATION WH
36
                                           -ERE THE IMAG WAS STORED.
37
38
39
        ; NOTE:
40
41
               NOW THE REST OF THE POINTS ARE PROCESSED, SO
42
               GET RM IM, RP AND IP FOR ALL EXCEPT THE FIRST
43
               POINT
```

```
1
         POST5:
                                              SINE=SINE(R5)
                  HOV
                            SINTAB(R5), SINE
 2
                  MOV
                           COSTAB(R5).COS
                                              ;COS=COS(R5)
 3
                  MOV
                            (R4), RP
                                              ;RP(N-M)
                  MOV
                            (R4),RM
                                              ;RM=R(N-M)
                  NEG
                           RM
                                              ;R11=-R(11-11)
                  MOV
                            2(R4), IP
                                              ; IP=I(N-M)
 7
                  MOA
                            2(R4), IM
                                              :IM=I(N-M)
 9
                  NEG
                           IM
                                              ;IM=-I(N-M)
 9
                  ADD
                            (RO) RP
                                              ;RP=R(M)+R(N-M)
10
                  ADD
                            (RO), RM
                                              RM=R(M)-R(N-M)
11
                  ADD
                           2(R0). IP
                                              ; IP=I(M)+I(N-M)
12
                  ADD
                           2(RO), III
                                              ; IM=I(M)-I(N-M)
13
         POST10: MOV
                           RP ARM
14
                  MOV
                           IM AIM
                                              ; ARM=AR(M) AND AIM=AI(M).
15
                  YOM
                           RP, ARNM
16
                  MOV
                           IM, AINM
                                             :ARNM=AR(N-M) AND
17
                                              ; AINM=AI(N-M).
18
                  NEG
                           AINM
                                             ;AINM=-AI(N-M).
19
                  MOV
                           IP R2
20
                  MUL
                           COS.R2
                                             ; IP*COS(T)
21
                  ASHC
                           #1 R2
22
         POST20: ADD
                           R2, ARM
                                             ARM=A(M)+IP*COS(T)
23
                  SUB
                           R2, ARNM
                                              \cdotARNM=AR(N-N)-IP*COS(T)
24
         POST25: MOV
                           IP,R2
25
                  MUL
                           SINE.R2
                                             ; IP*SIN(T)
26
                  ASHC
                           #1,R2
27
                  SUB
                           R2, AIM
                                             ;AIM=AI(H)-IP*SIN(T)
28
                  SUB
                           R2.AINM
                                             :AINI=AI(N-II)-IP*SIN(T).
29
                  MOV
                           RM R2
30
                  MUL
                           SINE, R2
                                             ; RM*SIN(T)
31
                  ASHC
                           #1 R2
32
                  SUB
                           R2.ARM
                                             ; ARM=AR(M)+IP*COS(T)-RM*SIN(T)
33
                  ADD
                           R2, ARNM
                                             ; ARNM=AR(N-M)-IP*COS(T):R11*SIN(T)
34
                  MOV
                           RM, R2
35
                  MUL
                           COS,R2
                                             ; RM*COS(T)
36
                  ASHC
                           #1,R2
37
         POST50: SUB
                           R2 AIM
                                             ;AIM=AI(M)-IP*SIN(T)-RM*COS(T9
38
                  ŜUB
                           R2, AINM
                                             ;AINM=AI(N-M)-IP*SIN(T)-RM*COS(T)
39
         POST55: MOV
                           ARM, (RO)
40
                 MOV
                           ARNM (R4)
41
                 MOV
                           AIM, 2(RO)
42
                 MOV
                          AINM, 2(R4)
```

```
; NOTE:
 2
               ONE PAIR OF POINTS HAS BEEN DONE, NOW
               GO ON TO NEXT PAIR OF POINTS.
                                         ; INCREMENT SINE INDEX
                ADD
                        INDX R5
                                         ;INC RO BY 2
                CMP
                        (R0)+, (R0)+
                                         ;DEC R4 BY 2
                CMP
                        -(R4), -(R4)
10
                DEC
                        R1
                                         ;R1<-R1-1
11
                BNE
                        POST5
                DEC
                        SFACTR
                                         DIVIDE DATA ARRAY BY 2
12
                RTS
13
                        PC
14
15
16
17
18
19
20
21
22
23
24
        ; END OF POST PROCESSING ALGORITHM.
```

SUBROUTINE SCALE

```
.SBTTL SUBROUTINE SCALE
 3
        ;
        ;
 5
 6
          1. INTRODUCTION TO SUBROUTINE "SCALE":
 7
             THIS SUBROUTINE CHACKS BITS 13-14 AND SHIFTS THE
 9
        ; WHOLE ARRAY TO THE RIGHT BY TWO BITS IF BIT-14 IS SET,
10
        ; AND SHIFTS THE ARRAY ONE BIT TO THE RIGHT IF BIT-13
11
        ; IS SET.
12
              BEFORE CALLING THIS SUBROUTINE IT SHOULD BE
13
        ; ENSURED THAT THE STARTING ADDRESS OF THE ARRAY
14
        ; IS LOADED IN RO AND THE LENGTH OF THE ARRAY IS LO
15
        ; -ADED IN R1. THE SUBROUTINE IS CALLED BY GIVING THE
16
          COICIAND "JSR PC SCALE .
              THE VARIABLE "SFACTR", WHICH IS DEFINED BELOW,
17
        ; KEEPS TRACK OF THE # OF TIMES THE ARRAY HAS BEEN
18
19
        ; SHIFTED TO THE RIGHT BY ONE BIT.
20
21
22
23
24
        .PSECT SFACTR, RW.D. GBL, REL, OVR
25
                SFACTR: 0
                                ; SFACTR IS DEFINED HERE.
26
        . PSECT
```

SUBROUTINE SCALE

49

```
; 2. CODE FOR SUBROUTINE SCALE
        SCALE: MOV
                         RO R5
                                          GET STARTING ADDRESS
                                          ·AND STORE IN REG R5.
                                          GET ARRAY LENGTH AND
                MOV
                        R1, R3
                                          STORE IN REG R3.
                CLR
                         R4
                                          CLEAR R4 WHICH IS A FLAG
 9
                                          TO KEEP TRACK WHETHER BIT
10
                                          :-14 OR BIT-13 IS SET.
        SCAL1: MOV
                         (R0)+,R2
                                          GET EACH POINT
11
12
                BPL
                         SCAL2
                                          ;SKIP NEXT INSTRUCTION IF
13
                                          ; VALUE IS POSITIVE.
14
                NEG
                        R2
                                          GET ABS VALUE
15
        SCAL2: BIT
                         #40000.R2
                                          ;IS BIT-14 SET?
16
                BNE
                        RDUCE2
                                          ;YES, SCALE TWICE.
                                          ;IS BIT-13 SET?
17
                BIT
                         #20000 R2
18
                BEQ
                         SCAL3
                                          :NO, GET ANOTHER POINT.
19
                MOV
                         #1.R4
                                          ;YES, SET FLAG AND CHECK
20
                                          ; NEXT POINT.
21
        SCAL3: SOB
                        R1,SCAL1
                                          ;ALL POINTS CHECKED?
22
                TST
                                          ;TEST IF SCALING IS REQUIRED.
                        R4
23
                BNE
                         RDUCE1
                                         ;YES IT IS REQUIRED.
24
                RTS PC
25
        ; NOTE:
26
27
                SHIFT ENTIRE ARRAY EITHER ONE OR TWO BITS
28
29
                TO THE RIGHT.
30
        RDUCE2: CLR
31
                        R4
32
                INC
                         SFACTR
33
        RDUCE1: INC
                         SFACTR
                TST
34
35
                BNE
                        RDUCE4
                                          SHIFT EACH POINT TWO
36
        RDUCE3: ASR
                         (R5)
                                          ;BITS TO THE RIGHT.
37
                         (R5)+
38
                ASR
39
                SOB
                        R3, RDUCE3
                                          ;ALL POINTS SCALED?
40
                RTS
                         PC
                                          :YES
        RDUCE4: ASR
41
                         (R5)+
                SOB
                         R3 RDUCE4
42
43
                RTS
                         PC
44
45
46
47
48
```

; END OF SCALING SUBROUTINE.

SUBROUTINE "SWAP"

```
.SBTTL SUBROUTINE "SWAP"
 3
          1. SUBROUTINE "SWAP" (INTRODUCTION AND CODE):
 8
              THIS SUBROUTINE IS USED TO SWAP THE CONTENTS OF
10
          MEMORY LOCATIONS POINTED TO BY REGISTERS RI AND R2.
11
        ; BEFORE CALLING THIS SUBROUTINE IT SHOULD BE ENSURED
        ; THAT THE ADDRESS OF MEMORY LOCATIONS THAT ARE TO BE
12
13
        ; SWAPPED ARE LOADED IN R1 AND R2.
14
15
16
17
        SWAP:
18
                VOM
                        (R1), R3
19
                MOV
                        (R2) (R1)
20
                MOV
                        R3, (R2)
21
                RTS
                        PC
22
23
24
25
26
27
28
29
30
31
        ; END OF SUBROUTINE SWAP.
```

SUBROUTINE "LOOKUP"

; N2 (128) WITH THE VALUES FOR THE SINES STARTING AT SINLOK,

; AND FOR COSINES STARTING AT COSLOK. EACH FUNCTION HAS 128

; ENTRIES (THE LAST HALF OF THE SINE TABLE IS THE FIRST HALF OF THE COSINE TABLE). THE "PASS2 AND ONWARD" SECTION USES

EVERY OTHER VALUE (K*PHI/(N2/2)) AND THE "POST PROCESSING" SECTION USES EVERY VALUE.

SUBROUTINE "LOOKUP"

```
; 2. LOOK UP TABLE
 1
 2
           ______
 3
 4
          SINE (K*PHI/128), K=1 2,
                                        128 STARTS HERE
 5
        SINTAB: 000000,001444,003110,004552,006214,007653,011310
 6
 7
                 012742,014371 016013,017432,021044,022450,024047
 8
                 025437 027021,030374,031737 033272 034615 036127
                 037427, 040716 042173, 043435, 044664, 046100, 047300
 0
                 050464, 051633, 052766, 054103, 055202, 056264, 057327
10
                 060354,061362 062350,063317,064246,065155,065044
11
                 066712,067537,070343,071125,071666,072405,073102
12
13
                 073554, 074204, 074612, 075175, 075535, 076052, 076344
14
                 076612,077035.077235,077412,077542,077647,077730
15
                 077766
16
        ; COSTNE LOOK-UP TABLE STARTS HERE
17
18
        COSTAB: 077717,077766,077730,077647,077542,077412
19
20
                 077235,077035,076612,076344,076052,075535,075175
21
                 074612,074204,073554,073102 072405,071666,071125
22
                 070343, 067537 066712, 066044, 065155, 064246 063317
                 062350, 061362 060354, 057327, 056264, 055202, 054103
23
24
                 052766, 051633, 050464 047300, 046100, 044664, 043435
                 042173.040716.037477.036127.034615.033272.031737
25
                 030374 027021, 025437, 024047, 022450, 021044, 017432
26
27
                 016013, 014371 012742, 011310, 007653, 006214 004552
28
                 003110,001444,000000
29
        ; END OF SINE LOOK-UP TABLE
30
31
32
                 176334, 174670, 173225, 171564, 170125, 166470, 165036
33
                 163407, 161764, 160346, 156734, 155330, 153231, 152341
34
                 150757, 147404, 146041, 144506, 143163, 141651, 140351
35
                 137062 135605, 134343, 133114, 131700, 130500, 127314
36
                 126145, 125012, 123675, 122575, 121514 120450, 117424
37
                 116416, 115427, 114460, 113531, 112622, 111734, 111066
38
                 110241, 107435, 106653, 106112, 105373 104676, 104224
39
                 103573, 103166, 102603, 102243, 101726, 101434, 101166
40
                 100742, 100543, 100366, 100236, 100131, 100047, 100012
41
                 100000
42
43
          END OF COSINE LOOK-UP TABLE
44
45
                 . END
```

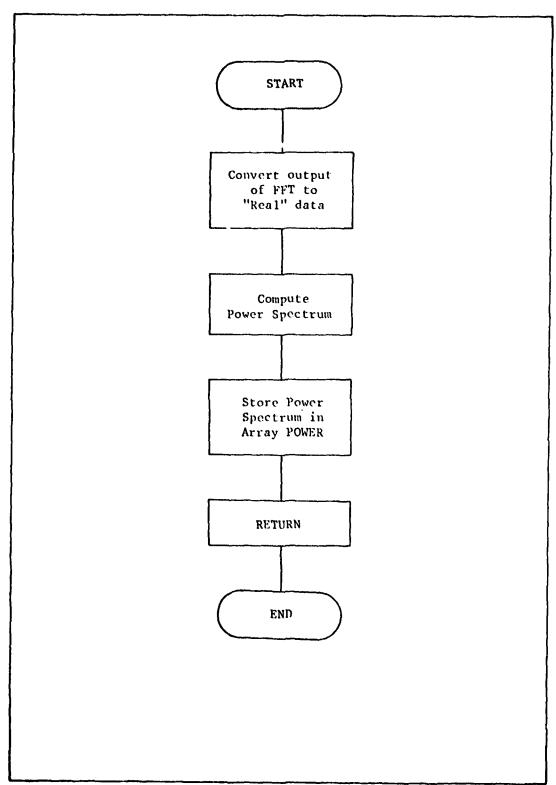


Fig 26. Flow Chart for POWERSPECTRUM

COMMON /ADDATA/ADATA,/SFACTR/SCALE

0005

(:

```
C
      C
            ARITHMETIC STATEMENT FUNCTION TO COMPUTE THE FLOATING POINT
      C
            VARIABLE EQUAL TO THE VALUE OF THE FIXED POINT FREQUENCY
      C
            SPECTRUM COMPUTED IN FOURIER TIMES THE GAIN OF THE LINEAR
      C
            AMPLIFIER.
0006
            FPDATA(I) = FLOAT(ADATA(I))*(2.**SCALE)*FPGAIN
         PROGRAM EXECUTION STARTS HERE.
0007
            FPGAIN=FLOAT(GAIN)
8000
            K=3
0009
            DO 100, I=1. NFREQ
0010
            POWER(I)=POWER(I)+FPDATA(K)**2+FPDATA(K+1)**2
0011
            K=K+2
0012
       100 CONTINUE
0013
            RETURN
0014
            END
```

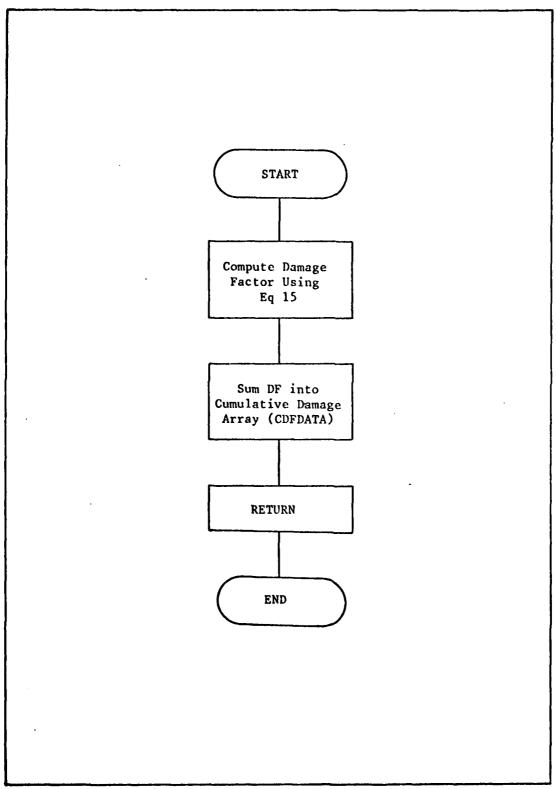


Fig 27. Flow Chart for DAMAGEFACTOR

```
FORTRAN IV
             VO2.1-11
                        Fri 23-Nov-79 12:47 09
                                                               PAGE 001
            SUBROUTINE DAMAGEFACTOR
0001
      C INTRODUCTION
      C THIS MODULE COMPUTES THE CUMULATIVE DAMAGE FACTOR (CDF)
      C BY SUMMING THE DAMAGE FACTORS. THE DAMAGE FACTORS ARE
      C COMPUTED USING AN ARITHMETIC STATEMENT FUNCTION -
      C DAMAGE (K, FREQUENCY) WHICH IMPLEMENTS EQUATION 15.
      C
      C
            VARIABLE DEFINITION
      C
      C LOCAL
                GLOBAL
                               DEFINITION
      C CDF
                CDFDATA
                         DATA THE ARRAY WHICH CONTAINS THE CUMULATIVE
                          DAMAGE FACTORS. I RANGES FROM 1 TO 127
      C
      C
                          AND TEMP RANGES FROM 1 TO 12.
      C
                         DELTA THE FREQUENCY INTERVAL
      C DELTAF DELTA
      C FREQUENCY
                          THE FREQUENCY AT WHICH THE CACULATION
                          IS TAKING PLACE.
      C
      C NFREQ
                         THE NUMBER OF FREQUENCY INTERVALS IN
                NINTVL
                          THE POWER AND CDF ARRAYS.
      C
      C
                PWRSPC
                          AN ARRAY WHICH CONTAINS THE POWER
      C POWER
                          SPECTRUM.
      C
      C TEMP
                TFLG
                          THE TEMPERATURE ARRAY IN CDF FOR WHICH
                          THE DATA TAKEN.
      C
      C
      C
         VARIABLE TYPE DEFINITION
      C
0002
            INTEGER*2 TEMP NFREQ
0003
            REAL*4 POWER (120), CDF (120.12) FREQUENCY, DELTAF
         GLOBAL VARIABLE DEFINITION
      C
      C
0004
            COMMON /PURSPC/POWER, /CDFDATA/CDF /TFLG/TEMP
0005
            COMMON /NINTVL/NFREQ,/DELTA/DELTAFV
            ARITHMETIC STATEMENT FUNCTION TO DETERMINE THE
      C
            DAMAGE FACTOR
0006
            DAMAGE(K, FREQUENCY) = ((POWER(K)**1.65)/(FREQUENCY**5.4))
0007
            FREQUENCY=DELTAF
8000
            DO 100, I=1. NFREQ
0009
            CDF(1.TEMP)=CDF(I.TEMP)+DAMAGE(I,FREQUENCY)
0010
            FREQUENCY=FREQUENCY+DELTAF
0011
       100 CONTINUE
0012
            RETURN
0013
            END
```

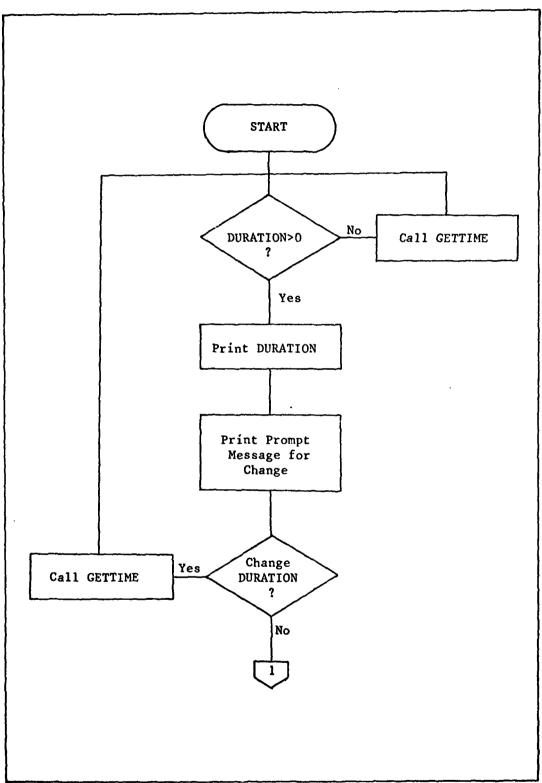


Fig 28. Flow Chart for XECUTE

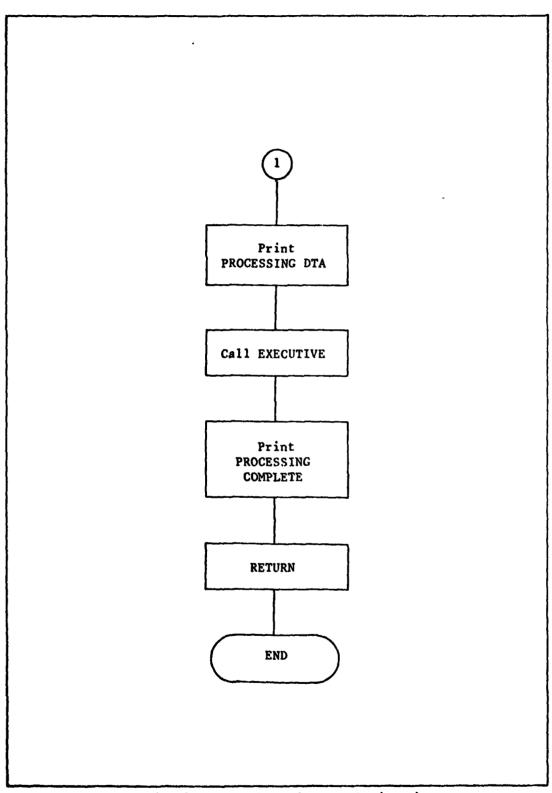


Fig 28. Flow Chart for XECUTE (cont)

```
FORTRAN IV
             V02.1-11 Fri 23-Nov-79 14:57:19
                                                             PAGE 001
0001
            SUBROUTINE XECUTE
      C
      C
      C
            IF THE TIME IS LESS THAN OR EQUAL TO ZERO, A NEW TIME IS
         OBTAINED BY CALLING SUBROUTINE TIME. AND THEN
      C
         THE DATA PROCESSING STARTS.
      C
      C
      C
         VARIABLES DEFINED HERE
      C
0002
            LOGICAL*1 ANSWER, ERR
0003
            INTEGER*2 MINUTES
0004
            REAL*4 DURATION
0005
            COMMON /PTIME/MINUTES
      C
        THE RUN TIME IS CHECKED AND IF IT IS GREATER THAN
       ZERO DATA PROCESSING IS STARTED. IF NOT A NEW TIME
      C
        IS REQUESTED.
      C
0006
            IF (MINUTES .LT. 0) CALL GETTIME (DURATION)
8000
       100 TYPE 200, MINUTES/60.
0009
       200 FORMAT(2X, 'THE DURATION IS '.F5.2 ' HOURS')
      C
         THE USER IS OFFERED ONE LAST CHANCE TO CHANGE THE RUN DURATION.
      C
0010
            TYPE 225
0011
       225 FORMAT (/2X, 'DO YOU WANT TO CHANGE IT ? Y/N<CR>'$)
0012
            ACCEPT 250, ANSWER
0013
       250 FORMAT (A1)
0014
            IF (ANSWER-EQ 'Y') CO TO 500
0016
            IF ((ANSWER.NE.'Y').AND.(ANSWER.NE.'N')) GO TO 600
0018
            TYPE 300
0019
       300 FORMAT (2X, 'DATA BEING PROCESSED')
0020
            CALL EXEC
0021
       399 TYPE 400
0022
       400 FORMAT (2X. 'DATA PROCESSING COMPLETE'/)
0023
            RETURN
0024
       500 CALL GETTIME (DURATION)
0025
            GO TO 100
0026
           CALL ERROR ('INPUT MUST BE A Y OR A N')
0027
            GO TO 100
0028
            END
```

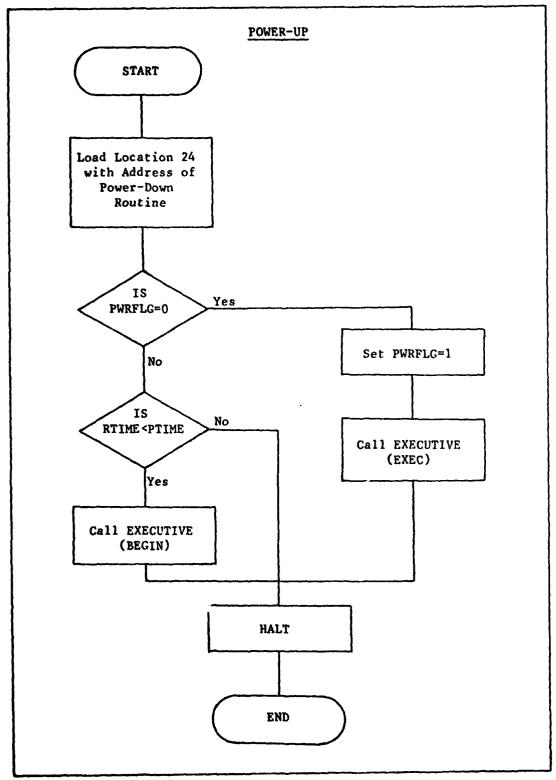
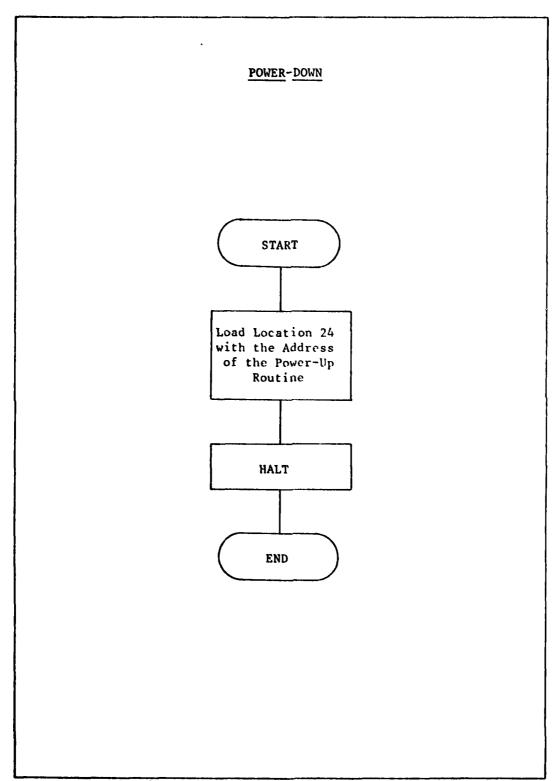


Fig 29. Flow Chart for POWERUPDN



ï

Fig 29. Flow Chart for POWERUPDN (Cont)

POWERUPDN PROCEDURE MACRO VO3.01 23-NOV-79 15 40:26 PAGE 1

```
.TITLE POWERUPDN PROCEDURE
 1
 2
     .SBTTL INTRODUCTION
 3
         INTRODUCTION:
     · 1.
 7
             THIS PROCEDURE IS USED WHEN THE AUTOMATIC STARTING MODE
 8
     ; IS SELECTED. WHEN SYSTEM POWER IS TURNED ON. WITH THE RUN/HALT
     ; SWITCH IN THE RUN POSITION (LSI-11 OPERATING IN MODE 0,) PROGRAM
10
     ; EXECUTION STARTS AT LABEL "PURUP".
                                            FIRST, THE ADDRESS OF THE
11
     ; POWER DOWN LABEL "PWRDN" IS STORED AT LOCATION 24. THEN THE
     ; GLOBAL VARIABLE "PWRFLG" IS CHECKED. IF IT IS EQUAL TO ZERO
12
     ; MEANING IT IS THE FIRST POWER-UP, "PURFLG" IS SET EQUAL TO
13
     ; ONE AND PROGRAM EXECUTION JUMPS TO THE LABEL "EXEC" IN THE
14
15
     ; EXECUTIVE MODULE IF "PWRFLG" DOES NOT EQUAL ZERO, THE RUN
16
     ; TIME IS CHECKED AND IF IT IS LESS THAN OR EQUAL TO THE PLANNED
17
     ; TIME PROGRAM EXECUTION JUMPS TO THE LABEL "BEGIN" IN THE
18
     ; EXECUTIVE MODULE. IF THE RUN TIME IS EQUAL TO OR GREATER THAN THE
     ; PLANNED RUN TIME PROGRAM EXECUTION STOPS.
19
20
21
22
     ; 2. GLOBAL VARIABLES:
23
24
25
     •GLOBL PWRFLG, EXEC, BEGIN, PTIME, RTIME
26
     .MCALL .REGDEF
27
     .REGDEF
28
29
30
     .PSECT PURFLG RW.D.GBL REL,OVR
31
     PWRFLG: .WORD 0
                             POWER FLAG USED TO DETERMINE IF IT IS THE
32
                              ;FIRST POWER-UP.
33
     • PSECT
34
35
36
     ;3. CODE FOR POWER-UP:
37
38
39
     PWRUP
             MOA
                     #PWRDN @#24
                                      ;SET UP VECTOR FOR THE
             MOV
40
                     #200,@#26
                                      ; POWER-DOWN ROUTINE.
41
             TST
                     PWRFLG
                                      ;CK IF IT IS THE FIRST
42
                                      ; POWER-UP; IF NOT.
43
             BNE
                     RESTRT
                                      ;CHECK THE RUN TIME
44
             MOV
                     #1 PWRFLG
                                      ; PWRFLG INDICATES THAT
45
                                      ;THE FIRST POWER-UP
46
                                      ; HAS OCCURRED.
47
             JSR
                     PC, EXEC
                                      ;START DATA PROCESSING
48
                                      ;AT LABEL "EXEC" IN THE
49
                                      EXECUTIVE MODULE
```

50 51 52 53 54	; MODUL ; HALT.			PROGRAM CONTROL TO THIS O PTIME SO PROGRAM WILL
55 56 57 58	ŘESTRT:	CMP BGE JSR	PTIME RTIME HLT PC, BEGIN	CK RUN TIME STOP IF RTIME=>PTIME OR ELSE FINISH PROCESSING THE DATA.
59 60 61 62 63	HLT:	HALT E FOR PO	WER DOWN MOD	
64 65 66 67 68	; PWRDN:	MOV HALT •END	#PWRUP.@#24	REPLACE THE POWER-DOWN VECTOR WITH THE POWER-;UP VECTOR AND STOP.

Appendix C

Users' Manual

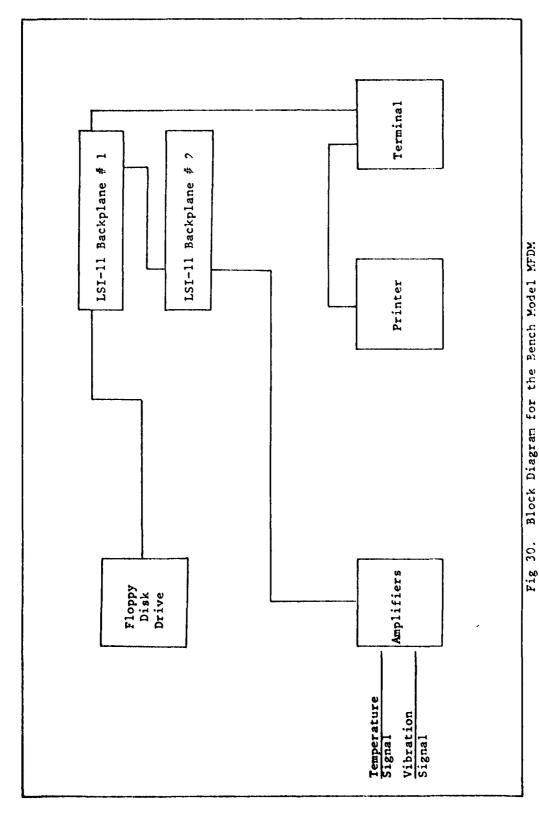
Introduction

The purpose of this manual is to explain how to use the existing bench model of the MFDM. First, the physical configuration of the system is discussed, and then its operation is discussed. The procedure for the automatic start-up mode is not discussed, since this mode of operation was not incorporated in the bench model.

Physical Configuration

The bench model system is designed to be used with a Lear Siegler ADM-3 terminal, a Heathkit H27 dual floppy disk drive and a Heathkit H14 printer. A block diagram for the system is shown in figure 30 with the backplane configuration shown in figure 31. The cable for the floppy disk must be connected to the disk interface module with the red dot on the connector on the top (visible). The bus expansion cable must be connected between the two backplanes with the M9400-YE module in the first backplane (with the processor board in it).

The band rate should be set at 4800 at the computer, the terminal, and the printer. A slower band rate can be used, if desired, as long as all three band rates are the same. The accelerometer is connected to the signal amplifier, with the output of the amplifier connected to the channel 16 leads of the A/D converter. Since the computer control of the auto ranging gain has not yet been implemented, the thumb wheel witch must be manually set to a value between 0 and 7; thus, manually setting the gain of the amplifier. The value to be set depends upon



*

1

The state of the state of

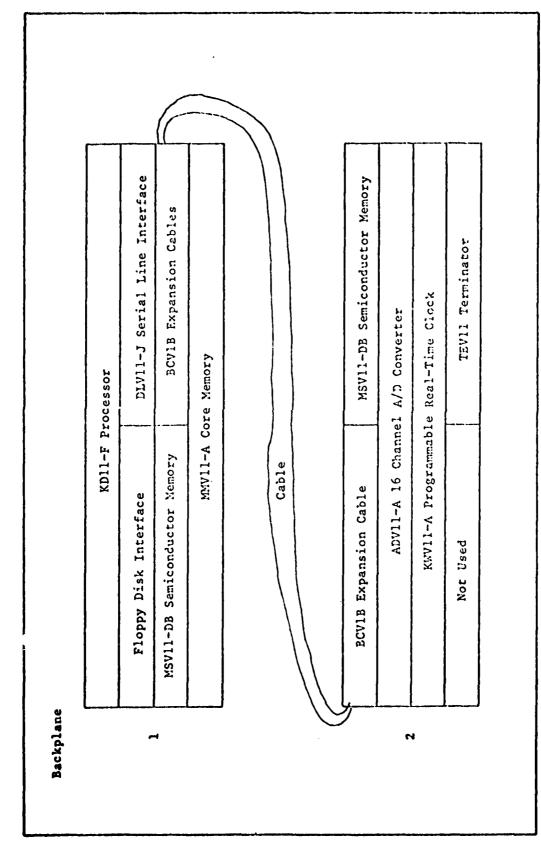


Fig 31. Bench Model Backplane Configuration

the intensity of the input signal. In order to determine what value to use, observe the output of the amplifier with an oscilloscope and set the switch to a position which keeps the magnitude of the strongest signal below 5 volts. The gain of the amplifier, as a function of the switch setting, is shown below.

Switch	Setting	Gain (db)
	0	-10
	1	0
	2	10
	3	20
	4	30
	5	40
	6	50
	7	60
	8	DO NOT
	9	USE

For the temperature input, the output of the temperature amplifier is connected to the channel 5 leads from the A/D converter (the thermocouple leads are permanently connected to the ampifier). This completes the connections for the vibration and temperature signals. The system is now ready to be turned on and to be configured for data acquisition and processing for the required frequency range and interval.

System Start-Up Procedure

First, the power should be turned on for all components (the computer, the terminal, the printer, and the amplifiers). After application of system power, the "RUN/HALT" switch should be placed in the "RUN" position, and then the "DC Power" switch should be turned on. A "\$" should appear on the terminal. If it does not check that all

units have power, and that all baud rates are set to the same value.

Insert a system disk with MFDM.SAV on it. in the left disk drive

(drive 0) and enter a "DX", followed by a carriage return, at the

terminal. If the system disk has a STARTF.COM file which contains the

instruction "R MFDM, then the system program will start executing

automatically. If not the following commands must be entered from the

terminal:

SET TT: FORM

SET TT:WIDTH=96

R MFDM

The first two commands configure the system to send form feeds to the terminal, and to set the length of a line to 96 characters. The last command starts the MFDM system program.

Operating the MFDM

The R MFDM command starts the execution of the MFDM system program. After starting, the user is greeted with the following lines:

MICROPROCESSOR FLIGHT DAMAGE MONITOR ENTER S(ETUP, E(XECUTE, O(UTPUT, Q(UIT - ?<CR>

The first letter of the function to be performed is entered, followed by a carriage return (<CR>), and that procedure is executed. The following sections describe each of the functions and their uses.

SETUP. If an "S" is entered, the setup procedure is executed.

This procedure allows the user to configure the system. The current

system configuration is printed at the terminal, along with the

following prompt line:

CHANGE D(ATE, C(OMMENTS, F(REQUENCY, R(UN, DURATION, V(ERIFY, Q(UIT - ?<CR>

The first letter of the desired procedure should be entered, followed by a carriage return, and that procedure will be executed. If a "V" is entered, the current system configuration will be displayed again, thus allowing the configuration to be checked. If a "Q" is entered, the set-up procedure will terminate, and the MFDM command line will be displayed.

<u>DATE</u>. If the date (which will be printed on the cumulative damage matrix table) is to be changed, a "D<CR>" is entered, and the following message is printed:

ENTER NEW DATE (13 characters max.) - ?<CR>

The new date is then entered in any form desired. For example these dates would be valid 15-DEC-79, 12-15-79, or 15 DECEMBER 79. However, the date must be no more than 13 characters long, and must end with a carriage return.

<u>COMMENTS</u>. By entering a "C<CR>", the comments can be changed. The following line is printed:

ENTER COMMENTS <CR>

Up to 40 printable characters can then be entered just as the user
wants them to appear on the output table followed by a carriage
return. These comments can be used for identification purposes such as
the flight number, test object and location.

FREQUENCY. When an "F<CR>" is entered, the following lines are printed:

FREQ.	RANGE (K Hz)	DELTAF (Hz)	ENTER
	0 - 6.0	100.2	1
	0 - 6.0	50-1	2
	0 - 3.0	100.2	3
	0 - 3.0	59.1	4
	0 - 3.0	25.0	5
	0 - 1.5	100.2	6
	0 - 1-5	50.1	7
	0 - 1.5	25 0	8

ENTER THE NUMBER CORRESPONDING TO THE DESIRED COMBINATION. ENTER AN INTEGER BETWEEN 1 AND 8 - ?<CR>

The frequency range for which the cumulative damage factor (CDF) will be obtained is 0 through the frequency selected, with the corresponding frequency interval - DELTAF. The frequency range and interval are initialized in the system by entering the integer (1 to 8) corresponding to the desired frequency range and intervals. Only the combinations shown above are available and any entry other than an integer between one and eight will result in an error message with a request for a new entry.

RUN DURATION. The run duration is the time for which data will be acquired and processed once execution begins. It can be set by entering an "R" followed by a carriage return, and then the desired duration should be entered. It should be entered in decimal hours, and followed by a carriage return.

Any of the four items can be changed, but all four do not have to be changed. When all of the desired changes have been made, entering a "Q" followed by a carriage return will terminate the set-up routine, and the MFDM command line will be printed as follows:

ENTER S(ETUP, E(XECUTE, O(UTPUT, Q(UIT - ?<CR>

This completes the SETUP procedure. The next procedure is EXECUTE

EXECUTE. The execute procedure is initialized from the MFDM command line by entering an "E" followed by a carriage return. The following message will then be printed:

THE RUN DURATION WILL BE XX HOURS DO YOU WANT TO CHANGE IT? Y/N<CR>

where the XX represents the current run time, which must be greater than zero. If the time needs to be changed, this is accomplished by entering a "Y" followed by a carriage return. The new duration should then be entered. If the time does not need changing, entering an "N" followed by a carriage return will result in the initiation of data processing. The following message will be printed at the terminal:

DATA BEING PROCESSED

Upon expiration of the run time, data processing will terminate, and the following lines will be printed:

DATA PROCESSING COMPLETE
ENTER S(ETUP, E(XECUTE, O(UTPUT, Q(UIT - ?<CR>

The data is now ready to be printed, and this is accomplished by the output routine.

<u>OUTPUT</u>. This routine prints out a table of the cumulative damage factor vs temperature for the frquency range selected, an example of which is shown in figure xx. The table is designed to be printed on a

96-column printer. In order to be able to print all of the data in 96 columns, the data has been divided into two categories depending on temperature. The first set of tables covers a temperature range of -50 degrees Fahrenheit to +100 degrees Fahrenheit, in 25 degree increments, printing 41 lines per page until all the frequency intervals have been printed. Then the printing process is repeated for a second temperature range of +75 degrees Fahrenheit to +225 degrees Fahrenheit.

After printing the CDF table, the MFDM command line is displayed, and the system is ready to repeat the test run using the same configuration (enter an "E"), to be re-configured for a new test (enter an "S"), or to print another copy of the CDF table (enter an "O").

Stopping Program Execution

When all of the testing has been performed, the system should be shut off. This is accomplished by removing the disk switching the DC power switch to the position, and then turning off all AC power off.

Summary

This completes the instructions on the operation of the MFDM bench model system. The procedures to configure the system and to run the programs have been discussed. If, for any reason, the system does not respond in the correct way, remove the disk shut down the power, and check the system configuration. Then re-start the MFDM. If the system still fails to operate check Chapters V-VII for the operation of the programs, in order to determine the problems.

Brian Charles Johnson was born July 24, 1948 in Minneapolis, Minnesota. He attended St. Louis Park High School in Minneapolis, graduating in 1966. He then attended the University of Minnesota, graduating in 1970 with a Bachelor of Physics Degree.

In 1971 he joined the U.S. Air Force, and received his navigator rating in 1972. He was stationed at Plattsburgh AFB, New York between 1972 and 1978, where he flew in the KC-135 aircraft. In 1978 he was selected to attend the Air Force Institute of Technology where he graduated with a Master of Science Degree (Computer Systems) on December 15, 1979.

Permanent Address:

7749 Abbott Ave. No. Minneapolis, Minnesota

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMEN	READ INSTRUCTIONS BEFORE COMPLETING FORM	
1. REPORT NUMBER	2. GOVT ACCESSION NO	
AFIT/GCS/EE/79-6	<u>.</u>	
4. TLTLE (and Subtitle)	1	5. TYPE OF REPORT & PERIOD COVERED
DESIGN AND IMPLEMENTA' MICROPROCESSOR FLIGHT	TION OF A	M.S. THESIS
MICROPROCESSOR FLIGHT	DAMAGE MONITOR	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(a)		8. CONTRACT OR GRANT NUMBER(s)
Brian C. Johnson Capt USAF		
9. PERFORMING ORGANIZATION NAME AN	D ADDRESS	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
Air Force Institute o		
Wright-Patterson AFB,	Ohio 45433 -	Project 7071-00-12
11. CONTROLLING OFFICE NAME AND ADD	DRESS	12. REPORT DATE
Structural Mechanics		December 1979
Air Force Flight Dynam		13. NUMBER OF PAGES
Wright-Patterson AFB,	Ohio 45433	15. SECURITY CLASS. (of this report)
THE MONITORING ACCITC FRANCE & ACCITCA	Salir directory from Southerning Southern	
		UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
Approved for public re	•	unlimited
17. DISTRIBUTION STATEMENT (of the absti	ract entered in Block 20, II different led	m Report)
18. SUPPLEMENTARY NOTES	Approved for public	release; IAW AFR 190-17
	JOSEPH D. HIPPS, Wa.	jor, USAF Affairs
19. KEY WORDS (Continue on teverse side if r	secassary and identify by block number	

Data Acquisition Signal Processing Microcomputer Fatigue Damage

20. ABSTRACT (Continue on reverse side if necessary and identity by block number)

In order to design a constrained layer damping treatment for aircraft components, the Air Force Flight Dynamics Laboratory at Wright-Patterson Air Force Base, Ohio requires a knowledge of vibration and temperature variations which a component encounters during its life. This report discusses the design and implementation of a microprocessor based system to acquire vibration and temperature information, and computes the cumulative damage factor

SECURITY CLASSIFICATION OF THIS PAGE(When Date Entered)

versus frequency and temperature in real-time. The cumulative damage factor is computed for one of eight user selectable frequency ranges and for a temperature range of -50 to +225 degrees Fahrenheit, and is stored in a non-volatile memory.

The designed system used an LSI-11 microcomputer system consisting of an LSI-11 processor, an analog-to-digital converter, a real-time clock, PROM, a non-volatile memory, and a serial interface. The software developed for this system controls the data acquisition, the computation of a fast Fourier transform (FFT), the power spectrum of vibration data, and the subsequent computation of the cumulative damage factor. The data acquisition and FFT modules were coded in assembly language, and the other modules were coded in FORTRAN. The output of the system consists of a table containing the cumulative damage as a function of frequency and temperature. A bench model system was constructed and tested, and a proposal for a flight-worthy system is made.